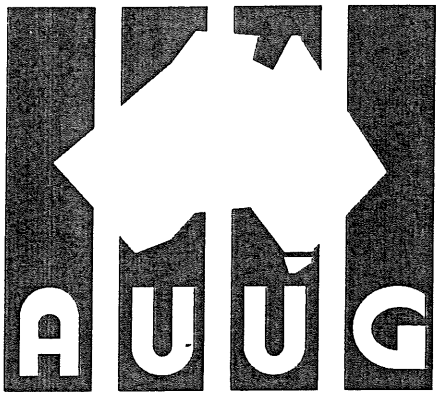
ISSN 1035-7521



Australian UNIX systems User Group Newsletter

AUUGN

Volume 12, Number 4/5

October 1991

Registered by Australia Post, Publication Number NBG6524

The AUUG Incorporated Newsletter

Volume 12 Number 4/5

October 1991

CONTENTS

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

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

The AUUG Incorporated Newsletter

Volume 12 Number 4/5

October 1991

CONTENTS

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

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

AUUG General Information

Memberships and Subscriptions

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

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
pzd30@juts.ccc.amdahl.com
Amdahl Pacific Services Pty. Ltd.
1 Pacific Highway
North Sydney NSW 2000

Vice-President

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

Secretary

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

Treasurer

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

Committee Members

Andrew Gollan
adjg@softway.sw.oz.au
Softway Pty. Ltd.
79 Myrtle Street
Chippendale NSW 2008

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

Peter Karr
Computer Magazine Publications
1/421 Cleveland Street
Redfern NSW 2016

Michael Tuke
mjt@anl.oz.au
ANL Ltd.
432 St. Kilda Road
Melbourne VIC 3004

Scott Merrilees
Sm@bhpese.oz.au
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

Firstly, I would like to apologise for the slight delay in publishing this issue (deadlines for contributions seem not to be taken too seriously). Note the deadline for the next issue given below, also in that issue I will publish the deadlines for 1992 issues of AUUGN.

Since the last issue we've had the AUUG'91 Conference, which I've been told was a great success, unfortunately I was unable to attend due to family commitments. Pat Duffy's comments on it in the Presidents report. Also included in this issue are the minutes of the Annual General Meeting, which took place at the conference, the financial statement and the Plan 9 paper as promised. A number of other local papers are also included in this issue together with some inclusion from affiliate organisations which will be of interest to AUUG members.

Book reviews have been started again, hopefully a number of reviews will appear in the next issue of AUUGN. Dave Newton (dave@teti.qhtours.oz.au) is the new book review editor. Anyone interested in reviewing books please contact Dave by e-mail or via the secretariat.

Finally, if you are working in the area of Open Systems, you surely come across interesting information for others. Why not write it up and submit it? I am always willing to help get it in a suitable format for printing.

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@munnari.oz.au*

Phone: +61 2 543 2552
Fax: +61 2 543 5097

Contributions

The Newsletter is published approximately every two months. The deadline for contributions for the next issue is Friday the 6th of December, 1991.

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 03/10/1991

(NSW) Department of Minerals & Energy
A.N.U.
AAII
Alcatel Australia
AIDC Ltd.
ANSTO
ANZ Banking Group/Global Technical Services
Adept Business Systems Pty Ltd
Adept Software
Apple Computer Australia
Apscore International Pty Ltd
Ausonics Pty Ltd
Australia Eds Pty Ltd
Australian Airlines Limited
Australian Bureau of Agricultural and Resource Economics
Australian Eagle Insurance Co. Ltd
Australian Electoral Commission
Australian Information Processing Centre Pty Ltd
Australian Taxation Office
Australian Wool Corporation
Avid Systems Pty Ltd
BHP CPD Research & Technology Centre
BHP Research - Melbourne Laboratories
Bain & Company
Ballarat Base Hospital
Burdett, Buckeridge & Young Ltd.
Bureau of Meteorology
Byrne & Davidson Holdings Pty Ltd
C.I.S.R.A.
CITEC
Codex Software Development Pty. Ltd.
Com Tech Communications
Commercial Dynamics
Communica Software Consultants
Computer Power Group
Computer Science of Australia Pty Ltd
Computer Software Packages
Corinthian Engineering Pty Ltd
CSIRO
Cyberscience Corporation Pty Ltd
DMP Software Pty Ltd
Data General Australia
Deakin University
Department of Transport
Dept. of Agricultural & Rural Affairs
Dept. of Conservation & Environment
Dept. of Defence
Dept. of Foreign Affairs & Trade
Dept. Of The Premier & Cabinet
Dept. of Treasury
Duesburys Information Technology Pty Ltd
ESRI Australia Pty Ltd
Eastek Pty Ltd
Emulex Australia Pty Ltd
Expert Solutions Australia
FGH Decision Support Systems Pty Ltd
Financial Network Services
First State Computing
Fremantle Port Authority
Fujitsu Australia Ltd
G. James Australia Pty Ltd
Genasys II Pty Ltd
General Automation Pty Ltd
George Moss Ltd
Hamersley Iron Pty. Limited
Harris & Sutherland Pty Ltd
IBM Australia Ltd
Iconix Pty Ltd
Infonetics
Internode Systems Pty Ltd
Ipec Management Services
Labtam Australia Pty Ltd
Leeds & Northrup Australia Pty. Ltd
Macquarie University
McDonnell Douglas Information Systems Pty Ltd
McIntosh Hamson Hoare Govett Ltd
Metal Trades Industry Association
Ministry of Housing & Construction (VIC)
Mitsui Computer Limited
Motorola Computer Systems
Multibase Pty Ltd
NEC Information Systems Australia Pty Ltd
OPSM
Oracle Systems Australia Pty Ltd

AUUG Institutional Members as at 03/10/1991

Parliament House
Prime Computer
Pulse Club Computers Pty Ltd
Q.H. Tours Limited
Queensland Department of Mines
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
Tasmania Bank
Tattersall Sweep Consultation
Telecom Australia
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
Turbosoft Pty Ltd
UCCQ
Unisys
University of New South Wales
University of Queensland
University of South Australia
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'91 has come and gone, and I guess none of you will be surprised if I devote a paragraph or two to the event.

Let's look at the exhibition first. By any standard, it was an overwhelming success, with more than 4500 people visiting the 65 exhibitors over the three days of the conference. We've received outstanding press coverage - The Australian, Canberra Times, Financial Review, Computerworld and Pacific Computer Weekly have all written about AUUG'91 in highly complimentary terms. In addition, we've featured in the first episode of the new ITVM - IT Video Magazine - which is out the week of Monday, October 14.

Sean Dent of PCW told me the other day that he's never seen a tradeshow like it. While I wouldn't go quite that far, I must admit, it reminded me of the halcyon mid-80's when a tradeshow such as the ACC had a "buzz", when every exhibitor was there because they knew they HAD to be there.

Now, why is any of this important? As I've said on numerous other occasions, for the conference to be a success - indeed, for it to have a future - the exhibition has to be a success. The exhibition underwrites all the activities associated with the conference and, most importantly, the two events are complementary, each contributing to the success of the other.

If we turn to the conference itself, while we didn't reach the 500 delegates hoped for, 411 was a pretty good number given the restrictions on travel and expenses that ALL companies have in place.

I must take this opportunity to recognise Andrew Gollan for his extremely hard work as Program Chair, and Piers Lauder for his outstanding contribution as a member of the Program Committee. There's nothing particularly glamourous about this job, which consists in large part of lots of administrative work, constant follow up, many meetings, and a lot of other tasks that seem pretty thankless, at least until the conference itself is underway. Clearly, we couldn't have a conference without a Program Committee and Program Chair, and I am mortified that I failed to emphasise this during the conference itself. I hope this mention in some small part makes up for that.

We had an excellent selection of speakers, both local and international, and a welter of topics to choose from. I also think we learned a lot of lessons this year, and I'd be interested in comments from any of you about the following:

1. I think we tried to pack too much into three days. In future, we should look at finishing the conference early on the third day, either at lunchtime, or with one final session after lunch, to allow interstate delegates to get away early. For ALL delegates, I think we're pretty well talked out by the end of two and a half days anyway.

2. I think we need to stream more effectively. That's easy to say from the vantage of hindsight, of course, considering that this was the first year that we've streamed at all. There's no doubt that streaming will be continued, and I think the goal is to more clearly position the technical and commercial topics so that an audience gets what it wants (and what it thinks it is going to get!).

3. I think there is scope to introduce a "management" stream - as opposed to general commercial - which would address strategic issues relating to open systems and would attract senior management. No matter how many improvements we've made, there's still little in the conference content to attract a CIO or IT Director or, for that matter, DP Manager from a large enterprise who needs to learn the benefits of open systems, how other organisations in the same industry are implementing open systems, and so on. There's

enormous pent up demand for this type of information in the market and who better than AUUG to provide it? (And, if we don't, someone else will.)

Further on that point, to be able to attract the high calibre of speakers we want, we must be assured that we can deliver to the speakers the audiences they deserve - in both quantity and quality. It's unfair to have a speaker make the effort to prepare a talk and, in many cases, travel thousands of miles to get here, to stand before 40 or 50 people who aren't really interested in that particular speaker's topic.

4. We must exercise more control over the content of presentations. Most of the presentations at AUUG'91 were excellent, but there were two or three that were blatant product pitches - and, in at least one example, not a particularly compelling product pitch at that. Obviously, we want to continue to have speakers provided by the major vendors, and it's legitimate for them to provide an update on their product developments, strategies, etc., but there are ways to do this without crossing over the line.

I spent a good deal of time with some of our overseas speakers to AUUG'91, notably Peter Cunningham, CEO of UI, Marie Burch, Director of International Operations, OSF, and John Totman, Director of User Council Relations, X/Open. They were fulsome in their praise of AUUG'91, both the conference and the exhibition, and all expressed the desire to return for AUUG'92. John Totman was overwhelmed at the size and professionalism of our event and felt that it outstripped anything he's seen in Europe.

Well, I guess that's enough on AUUG'91, although any comments, suggestions, recommendations, etc. from members are welcome. It's kind of a relief to be able to turn attention to something other than the conference for a while.

The next big issue to be tackled - two issues, really - is that of member benefits and state branches.

There is a Committee Meeting on October 25, and we will devote a large part of the day to those two issues. It's time for us to consolidate the progress made during 1991, take advantage of the financial base we've worked so hard to achieve, and serve our members in more creative and beneficial ways. I look forward to reporting some of our progress in the next issue.

AUUG
Minutes of Annual General Meeting 1991

Held at Darling Harbour Convention Centre, Sydney, 26th September 1991.

Office Bearers present: Pat Duffy (President), Chris Maltby (Vice President), Rolf Jester (Secretary), Frank Crawford (Treasurer).

Meeting commenced at 5:30 pm.

1. Minutes of previous Meeting.

Refer AUUGN December 1990. Moved by Richard Buckdale, seconded by Greg Kable, that the minutes from the last Annual General Meeting be accepted as printed. Carried.

2. President's Report

Pat Duffy reported that AUUG has begun a program of increased activity to serve the growing membership. The function of the Secretariat now having been given to a professional organisation (ACMS), the Committee has been able to start work on attracting new members for AUUG, to meet a real need in the market for what we as an organisation can provide, such as information.

We now have a proper membership database, and now write to welcome new members rather than just waiting to send them the next AUUGN. Renewals of membership have been streamlined to two renewal dates per year to simplify things for the member and to make follow-up easier. A survey in the Asia-Pacific Open Systems Review has been one of the sources of new membership applications.

AUUG now has formal links with X/Open, so that Australian users have a way of participating in the X/Open process of furthering the standards process. Similarly, our affiliation with UniForum gives our members access to the resources of that international user organisation.

Press activity through our PR agency has led to considerably increased visibility for AUUG, as part of our drive to position this organisation as a credible body for information on open systems.

At the end of the second day of AUUG'91, it is already clear that this event is an outstanding success. The continued vendor support will mean that we can look forward to continuing these popular conferences.

Pat concluded by saying that we shall also continue the PR campaign and general efforts to improve the quality of AUUG activities and publications. Amongst other things, we shall try to foster the formation of local user groups under the AUUG umbrella.

Moved by Stephen Prince, seconded by John Wright, that the President's Report be accepted. Carried.

3. Secretary's Report

Rolf Jester reported that the success of the increased activity mentioned by Pat Duffy was beginning to be seen in the increases in membership numbers and in the enormous numbers attending the exhibition this year. At the end of the second day, the numbers of visitors to AUUG'91 has already more than doubled the total for the whole of the 1990 event. A large part of the credit for this success goes to Wael Foda and

ACMS. On a continuing basis, the professional Secretariat services being provided by ACMS mean that AUUG can deliver a more responsive service to members.

The latest membership numbers, from August 1991, were 637, an increase of 45 over June 1991. There were 227 institutional members, 379 individual members and 21 subscriptions. Recent recruiting activity, not least at the AUUG stand at the exhibition, should see even more substantial growth.

Moved by Peter Chubb, seconded by Peter Alley, that the Secretary's Report be accepted. Carried.

4. Treasurer's Report

Frank Crawford submitted the following Income & Expense report and Balance Sheet.

attached

Frank reported that this year's costs were down largely due to the fact that we unfortunately published fewer AUUGNs.

Part of the cash balance will be transferred to a cash management or term deposit account to earn interest. We shall examine alternatives to ensure the most favourable return. But a sufficient balance must be maintained to underwrite activities like this AUUG'91 event and continuing and planned new services for members like a library.

The contribution from AUUG'91 looks like being around $60,000.

Frank concluded by stating that our liabilities are quite small, being mainly accrued expenses and pre-paid memberships.

Moved by Lawrence Brown, seconded by Peter Williams, that the Treasurer's Report be accepted. Carried.

5. Returning Officer's Report

John O'Brien reported as follows.

The last election was conducted from late June to early July. There were several nominations for various positions. Most of the nominations were withdrawn prior to the election so that most of those nominated were elected unopposed. The results are as follows.

President, Pat Duffy. Vice President, Chris Maltby. Secretary, Rolf Jester. Treasurer, Frank Crawford. Returning Officer, John O'Brien. No-one nominated for Assistant Returning Officer. The Committee has an opportunity to make an appointment to fill this position.

The Committee was another matter. An election was required. 74 formal ballot papers were received. There were 5 members to be elected. Nominees were Andrew Gollan, Peter Karr, Glenn Huxtable, Patrick McGrory, Scott Merrilees, Stephen Prince and Michael Tuke. The results were as follows.

Andrew Gollan and Glenn Huxtable received enough primary votes for immediate election. The other three positions were determined by preferences. They are Peter Karr, Scott Merrilees and Michel Tuke.

In addition to the election of office bearers there was a ballot about affiliating with UniForum. The Constitution says on Affiliation or Amalgamation with other organisations that the management committee may at any time seek or discuss the possibility of affiliation or amalgamation with any other organisation whose aims are similar to or compatible with those of AUUG. No agreement for affiliation or amalgamation may be finalised until the matter has received the assent of three quarters of the members voting in a postal ballot.

The reason for bringing these provisions to your attention was that the committee had discussed the benefits of affiliation with UniForum, the international association of UNIX system users. Key benefits

include:

- A 20 percent rebate of UniForum membership fees to AUUG for each member who joins or is currently a UniForum member,
- Discounts on volume purchases of UniForum publications,
- Discounts on UniForum conference registration,
- Assistance in recruiting speakers for meetings.

The most significant benefit of affiliation, in the opinion of the Committee, was that it brings AUUG closer to the international open systems community. The Committee recommended that the membership vote yes. And you did. Moved by Richard Burridge, seconded by John Wright, that the Returning Officer's Report be accepted. Carried.

6. Other Business

6.1 Assistant Returning Officer

It was suggested that Prof John Lions be co-opted as Assistant Returning Officer, and he indicated that he would accept that. The Committee agreed to put this on the agenda for the next Committee meeting for formal approval.

6.2 AARNET

Chris Maltby reported the success of our program of accepting AARNET address registrations. This has contributed nearly $10,000 to our net income.

However, it raises the issue of the need for advice as to how to actually connect to the network, which seems to be a need in the market that is not being met at present. Discussion at the meeting indicates that there is a need for a UUNET type of service.

Committee members expressed the opinion that AUUG itself is not really in a position to provide such a service, partly due to the investment required. However other service providers who already operate in related areas may wish to expand into this service. There was widespread support among those at the meeting for initiating a network service through an independent service provider.

The Committee agreed to discuss the idea with MHS and explore the possibility of encouraging and assisting them or another company in setting up a network service. Chris Maltby will follow up.

6.3 Software Distribution

Andrew McRae raised a suggestion that there be a Software Distribution at AUUG Conferences, in the form of a tape swap or CD mastering. General discussion indicated that this might be an additional attraction to join AUUG. Other ideas were that it could be part of the proposed AUUG library function (although then it would require staffing), that it be done via AARNET, that a machine be provided to make individual copies of tapes, or to encourage vendors to do that on their stands.

The idea will be pursued by the Committee for future Conferences.

6.4 Publications

Pat Duffy reported that AUUGN is now being published regularly again. The combined #4/5 issue will be out within a month of this Conference, and the last one for this year in December. Next year there will be six issue.

We also plan to publish a "glossy" magazine twice a year aimed at the present and potential commercial membership.

7. Next Meeting

At AUUG'92 Conference, World Congress Centre, Melbourne, September 1992. Exact date and time will be advised through AUUGN and Conference invitation.

The Meeting closed at 6:35 pm.

A.U.U.G. INCORPORATED

PROFIT & LOSS STATEMENT

FOR THE PERIOD 1ST JUNE, 1990 TO 31ST MAY, 1991.

	1991	1990
Conference A.U.U.G. 1990		
Income	34991.58	
LESS: Expenses		
Advertising		737.92
Photocopying & Printing	134.40	626.80
Travel & Accommodation	550.00	
Telephone	3.74	
	688.14	1364.72
NET PROFIT (LOSS)	34303.44	(1364.72)

A.U.U.G. INCORPORATED

PROFIT & LOSS STATEMENT

FOR THE PERIOD 1ST JUNE, 1990 TO 31ST MAY, 1991.

INCOME	1991	1990
Membership	40511.05	53190.60
Nutshell Handbooks		19930.15
Usenix Proceedings		
- Baltimore	398.00	330.00
- San Diego		492.00
A.U.U.G.N./Back issues	5757.00	4147.11
Subscriptions	1250.50	1461.00
Mailing List	3203.00	5959.50
Interest Received	7657.85	5431.13
Summer 90		
- Melbourne		3873.00
- Sydney		3149.95
- Canberra		250.00
Security Video		360.00
Security Pacific National Bank		509.24
Other Books	175.00	
Mugs	35.00	
Tutorial Notes	105.00	
Uniforum Rebate	599.94	
Summer 91		
- Melbourne	3884.20	
- Perth	2585.00	
AARNET Subscriptions	8858.00	
	75019.54	99083.68

LESS: Expenses	1991	1990
Bank Charges		
- Credit Card	535.63	440.76
- Government	31.20	127.26
- General	252.92	88.39
	819.75	656.41
MANAGEMENT COMMITTEE/MEETING EXPENSES		
- Airfares	4306.00	4325.50
- Accommodation/Meals	1497.35	485.70
- Parking	26.00	28.00
- Taxis	175.36	205.15
- Registration		44.00
- Postage	64.76	
- Fuel	175.97	
- Business Cards	490.80	
- Editors Float		200.00
	6736.24	5288.35
MEMBERSHIP		
- Freight/Postage		856.96
- Photocopying		16.80
- Printing	274.00	2396.93
- Product Directory		4117.34
- Secretarial Fees	722.00	
- Leaflets	150.00	
	1146.00	7388.03
UNIFORUM DELEGATION		
- Air Fares	2549.90	
NUTSHELL		
- Freight/Postage		2043.68
- Purchases Handbooks		6903.18
		8946.86
USENIX PROCEEDINGS		
- Baltimore		734.63
		734.63
A.U.U.G.N.		
- Postage/Freight	94.03	3444.88
- Printing	12191.61	33348.88
- Laser Toner	220.00	
- Labels	57.87	
	12563.51	36793.76

		1991	1990
SUMMER 90			
- Administration			2898.70
- Melbourne			3576.62
- Sydney			3337.20
- Tasmania			358.00
			10170.52
SUMMER 91			
Melbourne	- Postage	615.00	
	- Printing	192.74	
	- Labels	49.98	
Video Costs		835.00	
Workshop Expenses		294.00	
Perth	- Seminar Costs	560.00	
	- Function Room	1848.00	
		4394.72	
MAILING LIST			
- Photocopying/Printing		2362.91	667.20
- Labels		162.80	
		2525.71	667.20
PROMOTION & PUBLICITY		5000.00	
OFFICE			
- Auditors Remuneration		1896.00	1850.00
- Freight/Postage		495.37	1284.51
- Petty Cash			101.20
- Printing/Stationery		855.95	2379.92
- Trademark Registrations		100.00	25.26
- 89 & 90 Election Costs		1332.55	
- Telephone		125.68	
- Secretary Fees		800.00	
- Business Cards		330.61	
- Registration Fees (Corporate Affairs)		27.50	
- Taxis		54.50	
		6018.16	6108.89
TOTAL OPERATING COSTS		41753.99	76754.65
General A/C Net Profit (Loss)		33265.55	22329.03
A.U.U.G. 89 Net Profit (Loss)			21389.77
A.U.U.G. 90 Net Profit (Loss)		34303.44	(1364.72)
NET PROFIT		67568.99	42354.08

A.U.U.G INCORPORATED

BALANCE SHEET

AS AT 31ST MAY, 1991

	NOTE	1991	1990
ASSETS			
CURRENT ASSETS			
Cash		121047.51	25151.48
Receivables	(3)	5350.00	6168.50
Investments	(2) (4)	35497.00	58552.93
		161894.51	89872.91
NON-CURRENT ASSETS			
Intangibles		988.10	988.10
Total Non-Current Assets		988.10	988.10
TOTAL ASSETS		162882.61	90861.01
LIABILITIES & CAPITAL			
Current Liabilities			
Trade Creditors		4452.61	
ASSOCIATION FUNDS			
Accumulated Profits		158430.00	90861.01
TOTAL LIABILITIES & CAPITAL		162882.61	90861.01

A.U.U.G. INCORPORATED

NOTES TO AND FORMING PART OF THE ACCOUNTS

FOR THE YEAR ENDED 31ST MAY, 1991.

1. **ACCOUNTING POLICIES**

The accounts are prepared in accordance with the historical cost convention. The Accounting policies adopted are consistent with those of the previous year.

2. **INVESTMENTS**

Investments are shown at Market Value, Capital Gains Tax is not taken into account in determining the investments unless a definite decision to sell has been taken and the related Capital Gains Tax can be reliably estimated.

Dividends and other distributions from investments are taken to income on receivable basis.

3. CURRENT RECEIVABLES	1991	1990
TRADE DEBTORS	$	$
Membership	1300.00	1328.00
Mailing List		1133.50
Video		160.00
Nut Shell Books		3547.00
Newsletter	4050.00	
	5350.00	6168.50

4. CURRENT INVESTMENTS	1991	1990
QUOTED INVESTMENT		
Chase AMP	6000.00	31000.00
C.B.A	29497.00	27552.93
	35497.00	58552.93

5. Trade Creditors	4452.61	-

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
Trosmo	Spring'90	Graphics Workshop IV	Oct'87

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

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

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

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? If **yes**, approximately how much?	Yes	No	_____
A4.	Do you require assistance in setting up your network software?	Yes	No	
A5.	Type of software:	SUNIII TCP/IP Other (Please specify): _____	MHSnet SLIP	UUCP
A6.	Type of connection:	Direct X.25/Dialin Other (Please specify): _____	Modem/Dialin X.25/Dialout	Modem/Dialout
A7.	If **modem**, connection type:	V21 (300 baud) V22bis (2400) Other (Please specify): _____	V23 (1200/75) V32 (9600)	V22 (1200) Trailblazer
A8.	Estimated traffic volume (in KB/day): (not counting netnews)	< 1 > 100: estimated volume: _____	1-10	10-100
A9.	Do you require a news feed?	Yes Limited (Please specify): _____	No	
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)? If **yes**, approximately how much (please also specify units, *e.g. $X/MB* or flat fee)?	Yes	No	_____
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): (not counting netnews)
< 1 1-10 10-100
> 100: acceptable volume: _____

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

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

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

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

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

B10. Additional Comments:

Plan 9, A Distributed System

Dave Presotto
Rob Pike
Ken Thompson
Howard Trickey

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

Plan 9 is a computing environment physically distributed across many machines. The distribution itself is transparent to most programs giving both users and administrators wide latitude in configuring the topology of the environment. Two properties make this possible: a per process group name space and uniform access to all resources by representing them as files.

1. Introduction

Plan 9 is a general-purpose, multi-user, portable distributed system implemented on a variety of computers and networks. Because commands, libraries, and system calls are similar to those of the Unix operating system, it is possible to port many Unix programs to Plan 9 with little or no changes. A casual user would find little difference between the two systems.

What distinguishes Plan 9 is its organization. The goals of this organization were to reduce administration and to promote resource sharing. Our programming style was minimalism. We believe that a small number of well-chosen abstractions can, with much less code, provide most of the function of a larger system. This is the approach that made the Unix operating system so much smaller than its contemporaries such as Multics. In building Plan 9, we generalized proven ideas from the Unix operating system rather than add new untried concepts.

Plan 9 is divided along lines of service function. Diskless CPU servers concentrate computing power into large multiprocessors; file servers provide repositories for storage; and terminals give each user of the system a dedicated computer with bitmap screen and mouse on which to run a window system. The sharing of computing and file storage services provides a sense of community for a group of programmers, amortizes costs, and centralizes and hence simplifies management and administration.

Since both CPU servers and terminals use the same kernel, users may choose whether to run programs locally on their terminals or remotely on CPU servers. Plan 9 provides this flexibility without constraining the choice. Therefore, both users and administrators can configure their environment to be as distributed or centralized as they wish. At work, users tend to use their terminals more like workstations running all interactive programs locally and reserving the CPU servers for data or compute intensive jobs such as compiling and computing chess end games. At home, connected via a dedicated 9600 baud line to work, users choose what they run locally and remotely to reduce communication cost. Some applications, such as the editor [Pik87], are split into multiple programs to make this choice even more flexible.

Figure 1 in any Plan 9 paper shows how we have configured our environment. Multiprocessor CPU and file servers are clustered in a few computer rooms and connected via 7 megabyte/sec point-to-point links [Pre88]. This permits the CPU servers to be used as high performance compute engines without becoming starved for data. Terminals are connected to the servers via lower speed, lower cost distribution networks such as the 10 megabit Ethernet [Met80] and 2 megabit Incon [Kal, Res]. By emphasizing the shared service clusters we can quickly and cheaply incorporate new technologies as they arise. At the same

time, users wishing more autonomy can incorporate as much computing power as they wish in their own offices without losing the advantage of transparently sharing other resources.

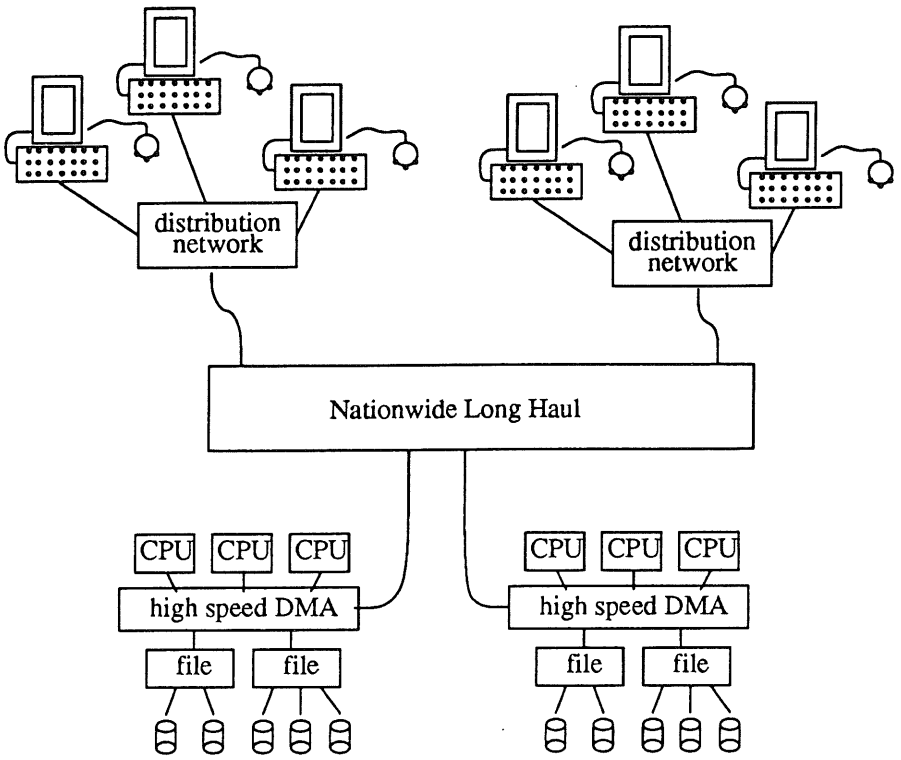


Figure 1 - Plan 9 Topology

The rest of this paper describes the features of Plan 9 that make possible such a flexible topology. For more information on hardware and use of the system, see our previous paper [Pik90] . For details of the file server, see [Qui] .

2. Minimalism

All resources that a process can access, aside from program memory, reside in one name space and are accessed uniformly. Simply stated, all resources are implemented to look like file systems and, henceforth, we shall call them file systems. A file system is a strict tree with no links. File systems can be the traditional type representing persistent storage on a disk as implemented by the shared file servers. They can also represent physical devices such as terminals or complex abstractions such as processes. The file systems can be implemented by kernel resident drivers, by user level processes, or by remote servers.

A file system representing a physical device normally contains one or two files. For example, an RS232 line is represented as a directory containing a `data` and a `ctl` file. The `data` file is the stream of bytes transmitted/received on the line. The `ctl` file is a control channel used to change device parameters such as baud rate.†

Some file systems represent software concepts. Environment variables (as in Unix) are implemented as files in a kernel resident file system. Even processes themselves are represented as directories with separate files representing different aspects of the process such as memory, text file, and control. Many things that require a system call in other operating systems are represented by I/O operations on files in Plan 9;

† We neither need nor have an `ioctl` system call.

reading the id of a process, the user id associated with a process, the time, etc.

A kernel data structure, called a *channel*, is used as a pointer to a file. A user level file descriptor is just a handle for a kernel channel. All I/O system calls eventually translate into nine primitive operations on channels. They are:

attach – point a channel to the root of a file system. The file system is told which user is attaching.

clone – make a copy of a channel. The new channel points to the same file as the old one.

walk – do a one level directory lookup on the channel and point it to the new file (or directory).

stat – get the attributes of the file pointed to.

wstat – change the attributes of the file pointed to.

open – check permissions prior to I/O on the channel.

read – read from the opened file.

write – write to the opened file.

close – close the opened file.

Each kernel resident file system is implemented by a *device driver* containing a procedure for each primitive operation. The device drivers are accessed indirectly via a kernel array, `devtab`, which contains 9 pointers per driver, one to each primitive procedure. Each channel contains an offset into `devtab` indicating the driver to be used in accessing the file it points to.

Accessing file systems not resident in the kernel is via a special device driver, the *mount driver*. All channels pointing to this driver contain a pointer to a communication channel. The mount driver turns operations on such channels into request messages written to the communication channel. The mount driver is written as a multiplexor allowing multiple outstanding messages. Because the messages on the communication channel are transmitted using `read's` and `write's`, any type of channel can be used: a pipe to a process, a network connection, even an RS232 line. The `mount` system call, described below, is used to create a new mount device channel and supply a communication channel for it.

All Plan 9 components are connected using this file system protocol. The code used to encapsulate the primitives into request and reply messages is 580 lines long. The mount driver is 899 lines long. Compared to the equivalent NFS code implementing vnodes and XDR this is tiny.

Of the 18000 lines of code that make up Plan 9, about 5000 lines perform memory management, process management, hardware interface, and system calls. The rest are for the 17 different file systems implementing devices, networks, process control, etc. Since most of the file systems are completely self contained, the complexity of the kernel code is even lower than its 18000 lines would imply. A working, albeit not very useful, kernel can be configured containing only the file systems implementing pipes, a local root, and a console. This totals 5899 lines of commented C code (counted using `wc *.[ch]`). As a comparison, Mach's micro-kernel without device drivers has 25530 lines of C code (calculated, we're told, by counting semi-colons). By the same metric our minimal kernel is only 4622 lines long, less than 1/5 the size. In fact, our kernel with every file system included is still less than half the size of their micro-kernel.

One might note the similarities between `devtab` and parts of the Unix operating system; the block device switch, character device switch, file system switch and vnodes. One advantage of Plan 9 is that we have recognized that these are all essentially the same mechanism and have implemented them as such.

3. Virtual Name Space

When a user boots a terminal or connects to a cpu server, a new process group is created for her processes. This process group starts with an initial name space that provides at minimum a root (`/`), some binaries for the processor the process is running on (`/bin/*`), and some local devices (`/dev/*`). The processes in the group can then either add to or rearrange their name space using two systems calls, `mount` and `bind`. The mount call is used to attach a new (not kernel resident) file system to a point in the name space. Its syntax is

```
mount(int fd, char *old, int flags, ...)
```

where *fd* is a file descriptor for a communication stream such as a pipe or a network connection and *old* is

the name of an existing file in the current name space where the file system will be attached. The attachment creates a new mount device channel whose communication channel is that referred to by *fd*. Subsequent accesses to *old* and any files below it in the hierarchy become request messages written to the communication stream.

The bind call is used to attach a kernel resident file system to the name space and also to rearrange pieces of the name space. Its syntax is

```
bind(char *new, char *old, int flags)
```

where *new* is a name in the current name space† and *old* is the same as in mount.

How the attachment works depends on the *flags* specified in the call. One possibility is that the old file is replaced by the new one. However, when both files are directories, Plan 9 allows another possibility. The result can be the union of the two directories. The effect is that of putting one directory behind the other. In the case of name conflicts for files contained in the directories, the one in front wins. *Flags* specifies whether the new directory replaces, goes in front of, or goes behind the old one. This concept is essentially the same as the search paths used in the Unix libraries and the various shells. In fact, Plan 9 has no search paths and uses these *union directories* in their place. When a command is executed, Plan 9 uses the directory `/bin` the same way Unix uses an execution path.

The ability to specify the complete name space for a process that contains all resources the process can access forms the basis for a true virtual machine. Any aspect of a process' world can be rearranged. Remote objects can be substituted for local ones. Processes can implement part or all of the name space of other processes. This capability is the basis for a number of important services, three of which we present here.

3.1. The Cpu Command

We consider the shared CPU servers as accelerators for our terminals, someplace where commands can run while maintaining the same environment. It is important that as little as possible change when running on the CPU server. The virtual name space provides us with a means to make the CPU servers actually feel this way to our users. A command, `cpu`, calls a CPU server across a network. A daemon process on the server answers the call, creates a new process group for the caller, sets up a name space, and starts a shell process in the new process group. The name space set up is an analogue of the name space of the calling process on the terminal. In particular, local resources on the terminal, such as the screen and the mouse, become visible to the server processes at the same place in the name space as on the terminal. The standard input, standard output, standard error, and current directory of the cpu command become those of the remote shell. The directories mounted on `/bin` are changed to be those that contain executables for the CPU server's processor type (the terminal may be a 68020 while a CPU server could be a MIPS). In general, a user typing the cpu command just notices that things such as compilations speed up while graphics operations slow down.

After the initial handshake to pass information describing the caller's environment, the cpu command becomes a file server answering file system requests from the network connection. The server daemon mounts the network connection to the terminal in a standard place, `/mnt/term`, and then binds the resources it decides to keep into the same places in the new name space. For example, it binds `/mnt/term/dev/mouse` onto `/dev/mouse`, `/mnt/term/dev/bitblt` onto `/dev/bitblt`, etc. Subsequent accesses to those files are converted by the mount driver in the CPU server into file system messages sent to the terminal.

† Local kernel resources are referred to by a syntactic escape (hack) in the name space. Any name starting with a "#" refers to a local resource. The first character following the "#" specifies the type of resource and the remaining characters are a parameter specifying the instance of the resource. Thus, to bind the local console to a standard place in the name space, one would use `bind("#c", "/dev", FRONT)`.

3.2. The Window System

The user interface is made up of three files:

`/dev/bitblt` – writes represent bitblt operations to the screen

`/dev/mouse` – reads return mouse events, i.e., button clicks and movement

`/dev/cons` – reads return keyboard input, writes put characters to the screen.

Between them, these devices represent all I/O to the user. The window system, 8.5 [Pik91], offers processes a multiplexed view to these devices. When a window is opened, the window system starts a new process group for a command (usually a shell) that will run in that window. In that process group's name space, the window system mounts a pipe to itself in front of `/dev`. Subsequent references by the new process group to any of these devices are sent as file system messages to the window server. 8.5 interprets those requests as accesses of the window instead of the whole screen. Similarly, 8.5 multiplexes the mouse and the keyboard so that mouse and keyboard input is available to processes only when their window is selected.

The result is that any program written to use the kernel resident user interface will also work inside a window. Because this is also true of the window system itself, new versions of the window system can be run and debugged in windows of the current window system.

3.3. Network Gateways

One, sometimes insurmountable, problem is accessing a network to which a system is not physically attached. For example, a system may be connected to our Datakit [Fra80] network but not to the DoD Internet. Many gateways exist that try to solve this problem by performing protocol to protocol translation. Unfortunately, few transport protocols have completely equivalent concepts. In order to perform the best translation, it is be necessary to know the semantics requested by the program. For example, TP4 has message delimiters but TCP does not. A protocol translator going from TCP to TP4 would not know which bytes correspond to a single write by the sender.

In Plan 9, every network interface is a file system. A gateway is a file server that serves its own network interfaces to other machines. A process that wants to get at a remote network connects to the gateway and mounts the gateway's interface to the remote network into its name space. Whenever the process accesses the interface, the mount driver will send the request to the gateway. Thus, the gateway sees exactly what the process does.

4. File Caching

In building our environment, we've been reluctant to add local disk file systems to any of our terminals or CPU servers. There are essentially two reasons for this choice. The first is administration. Anyone with a local disk must administer it. Any disk that has unique long term state requires both knowledge and time to administer. In fact, the Bell Labs computer center at Murray Hill is doing a lucrative business maintaining other peoples' disked Sun workstations because the owners have neither the time nor the experience necessary to do it themselves.

The second reason is sharing. Although most workstations can export access to their local file systems, when left up to individual users, this rarely happens. Terminals become personified and users become tied to a particular room to do their work.

Plan 9 survives without local disk file systems thanks partially to hardware and partially to caching. The CPU servers do so because their links to the file servers transfer at a substantial percentage of memory speed. The file servers maintain large main memory caches for their disk file systems. These servers are configured with 128 megabytes or more of main memory to ensure that there is plenty of room for cache. Getting a file from a file server is generally faster than it would be to get it from a local disk.

Office terminals are connected to the file servers by shared 1 or 10 megabit/sec links. Home terminals use 9600 or 19200 baud links. In both cases, the link is much slower than access to a local disk would be. To avoid the obvious performance hit, we use caching. To keep the caches coherent, we use file identifiers supplied by the file server. The identifiers are unique 64 bit quantities. 32 bits identify the file, the other 32 bits identify the version of the file. The version number is incremented each time the file is

modified. Each time a file is opened the file server returns the identifier with the reply. Therefore, it is possible to guarantee coherency at each opening of a file.

Office terminals only cache pages of executable files. Whenever a program terminates, its unmodified text and data pages are not immediately freed. Instead they are retained until the space is required by other programs. When a program is rerun its executable file is reopened and the current version number returned. If the version number has not changed and pages remain from the last run, they are reused. If the version number has changed, any remaining pages of the stale version are discarded. Since most data intensive work is done on the CPU servers, this simple cache saves most of the traffic between office terminals and the file servers. Other caching could be helpful but would require much more complexity.

This cache might also have sufficed for home terminals if it were persistent, but it is not. Therefore, we have added disks to our home terminals to be used as write through caches of the file server files. As a write through cache, it contains no state that isn't duplicated on the file servers. Therefore, it needs little maintenance compared to a local file system. If the code discovers a disk problem, it reformats the disk discarding the current contents. If the user should suspect that the cache is contaminated, she can request that it be reformatted at the next boot. The system slows down until subsequent use refills the cache but no information is lost. The user need not consciously update the disk because the cache uses file identifiers to maintain coherency with the file servers. Each time a file is opened, the cache discards any stale data it might have for that file. The user doesn't have to copy what she needs to the disk because it is done as a consequence of her using the data.

The disk based cache is implemented by a process that resides between the kernel and the file server connection. For every read request, the process satisfies as much as it can with data cached on the disk. It gets the rest from the file server. Any new data that passes through it is saved on the disk. When the cache fills up the least recently used file is discarded. The amount of data cached for any one file is limited to 1.75 megabytes to prevent one file from displacing all others.

Because the disk based cache only checks for coherency when a file is opened, it provides slightly different semantics than that seen on office terminals which do not cache data files. This looser coherency constraint forces programs that communicate via files to ensure an open between each transaction. Thus far we have not had to change any programs because of it.

5. Conclusion

We have presented a distributed system that is simple in structure and flexible in its use. Both the flexibility and simplicity are the result of two properties, a per process group name space and a single resource interface. Coupled with some minimal caching we provide a simple system that is as usable at home as at work.

6. Acknowledgements

Many people helped build the system. We would like especially to thank Bart Locanthi, who built our terminal, the Gnot, and encouraged us to program it; Tom Duff, who wrote the command interpreter `rc`; Tom Killian, who built and programmed the Gnot's SCSI interface; Ted Kowalski, who cheerfully endured early versions of the software; and Dennis Ritchie, who frequently provided us with much-needed wisdom.

References

Fra80. A. G. Fraser, "Datakit–A Modular Network for Synchronous and Asynchronous Traffic," in *Proc. Int. Conf. on Commun.*, Boston, MA (June 1980).

Kal. C. R. Kalmanek, "INCON: Network Maintenance and Privacy," Internal Memorandum 220106-0450, AT&T Bell Laboratories. .

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).

Pik91. Pike, R., "8.5, The Plan 9 Window System," *1991 USENIX Summer Conference Proceedings* (1991).

Pik87. Rob Pike, "The Text Editor sam," *Software - Practice and Experience* 17(11), pp. 813-845 (November 1987).

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).

Qui. S. Quinlan, "A Cached WORM File System," *Software – Practice and Experience*, p. To appear.

Res. R. C. Restrick, "INCON Wire Interface Integrated Circuit Design," Internal Memorandum 52413-860314-01TM, AT&T Bell Laboratories.

Realtime UNIX: Fact or Fiction?

Steve Kues

Concurrent Computer Corporation
11-15 Waverley Road, East Malvern VIC, 3145

ABSTRACT

UNIX[1] has come a long way since its inception at AT&T Bell Laboratories in the early 1970's. Today it is gaining great market acceptance by being able to provide an operating system that is versatile, portable, scalable and most importantly, vendor independent. UNIX also provides a rich set of development tools which makes it a favourite in both the commercial an scientific worlds alike.

One market still dominated by proprietary systems which UNIX has failed to penetrate in a major fashion is realtime computing. But what is realtime computing? Put most simply, a realtime system is one that can respond in a predictable and deterministic manner given impetus from the real world. These events must usually be serviced in a limited amount of time.

Just how close is UNIX to being seriously considered as a realtime operating system contender? What additions or changes need to be incorporated?

The aim of this paper is to discuss the realm of UNIX realtime computing by comparing and contrasting some of the realtime features available in various flavours of UNIX. Most of these features must be implemented through the kernel. The impact of these changes will be covered where appropriate. Defining a system which still permits normal operations of UNIX is a prime consideration.

Realtime Computing

Just what is *realtime* computing anyway. Many people use the term to describe their systems, while what they are really trying to say is: "I have a computer which really runs quite fast". Realtime has nothing to do with processor speed (although it does help implement good realtime systems), the amount of disk space you have, or memory to run your programs in. The whole aim of a realtime system is to be able to respond to a discrete event in a finite amount of time. This response interval must be directly calculable and bounded. Other cornerstones of realtime are consistency and reliability. A sequence of events must be reproducible, and data acquired quickly without compromising integrity. *Realfast* is not *realtime*. If you have a 100 MIP machine with a 1 MIP bottleneck, you have at best a 1 MIP machine.

In discussing the facets of realtime systems, I will draw upon the scenario of a fictional conventional UNIX operating system running in a similarly fictional hospital. I would have used a fictional nuclear power plant, although this analogy has become hackneyed, and Victoria is a nuclear-free state. I will use the term *normal* and *conventional* when discussing UNIX systems. This refers to an AT&T System V.3 or BSD 4.2 type system, that is, one not built for realtime operation.

Realtime systems may be implemented solely in software, although in order to achieve significant performance gains and reduce response intervals, it is usually necessary to include some hardware changes in the overall design.

---

[1] UNIX is a trademark of AT&T Bell Laboratories.

Hardware Considerations

In small realtime scenarios, where the load is not excessive, it is possible to implement a *satisfactory* system employing a single processor system. By "satisfactory", I am refering to the customer's definition, not the manufacturer's. If a patient connected to a computer monitoring device suffers a cardiac arrest, the operating system must warn medical staff of the arresting patient in a finite amount of time no matter how many orderlies are currently running *rogue* on the system. (Remember, this hospital is manned by fictional staff). On a normal UNIX system, there is no guarantee of how long before the software can respond to the interrupt. This is due in part to the fact that if only one processor is running all tasks, it must finish whatever activities it is currently doing (for instance, saving a *highscore*) before it can schedule a task to post a warning, or ring an alarm bell. Having multiple processors may not necessarily provide 100% guarantee, since there may be so many copies of *rogue* running that all installed processors are saturated.

Multiple processors will help in the following manner. Firstly, they provide additional computing resources. There are more CPU cycles available to application programs. This increases the chance of a processor becoming available in a smaller amount of time[2]. In general multiple processors give you the benefits of solving numerical parallel solutions more quickly through vectorisation, double buffering and pipelining of data acquisition systems. Probably the most important reason for having additional CPU's in our example is if the operating system supports slave CPU's. This is where a processor can be dedicated to a single (or group) of tasks. No other tasks can be scheduled on these CPU's, other than those defined. By similar means, all external interrupts can be tied to a single processor, offloading the *housekeeping* duties from other processors. The action-reaction time can be severely reduced.

The rate a normal operating system will task switch is 60Hz. This means at worst, the processor will be able to switch between 60 tasks per second. In most situations however, tasks will voluntarily relinquish the CPU through kernel calls which require resources (for example, waiting for disk I/O). If a computationally intensive application is running (that is, one that does not make kernel calls), it will not get preempted until the end of its quantum.

Time quantums are measured in multiples of system clock *ticks*, where a tick is usually 1000/60 = 16.7 milliseconds. This means that a task which does not relinquish CPU can at worst run a full 16.7 milliseconds before preemption. Allowing timers to interrupt at a faster rate will provide finer granularity in realtime applications. Rates of 120Hz or higher could be implemented, but there is a point of diminishing return where the processor can be interrupting tasks at such a rate they cannot get anything done.

[2]This assumes the operating system used supports task migration between all processors. To provide speed, global memory should also be implemented.

Software Considerations

Most of the proposals put forward here form part of the UNIX operating system on various manufacturers' implementations. Many of these are in addition being discussed by the POSIX realtime committee[3] which sets the standards. This committee is chaired by representatives of the manufacturers who have implemented realtime UNIX features. It should also be noted that almost all the following considerations must be implemented at the kernel level. User code almost invariably remains the same, for as mentioned earlier, a prime objective is for compatibility between standard flavours of UNIX.

Scheduling

The first mechanism which needs to be changed is the UNIX scheduler. I will not delve into the semantics of the System V.4 schedulers since I have not seen performance benchmarks on the effectiveness of their realtime scheduling. Instead I will focus on the standard UNIX scheduler found in System V.3 and BSD. This scheduler is orientated towards a timeshare system. It has been designed as the *fair scheduler,* providing a reasonable response to all users. The more CPU tasks use, the less response they receive from the processor. This enables commands which execute reasonably quickly to do just that. Tasks are said to *degrade* over time.

In a realtime environment however, this is clearly not acceptable. It is not acceptable for a task controlling a medicine dispensing machine to delay patients receiving their doses because the task has been running for an extended period.

In addtion, realtime priorities must be exactly that, prioritised. This means that if higher priority realtime task becomes ready to run, a lower task is preempted. After the higher level task completes, the lower level task is rescheduled. Critical care must be administered prior to general care for instance.

Many vendors now offer realtime schedulers which provide non-degrading priorities. It should be noted here that the *nice(2)* facility is not a satisfactory solution since although it raises and lowers priorities, the nice value is only one of a number of parameters consulted in the scheduling equation[4].

Allowing tasks to run with non-degrading priorities is not sufficient. Further enhancements need to be included, namely a choice of *timeslice* or *quantum*. This is the period of time a task will be allowed to run before preemption. If the execution time of a critical region can be measured exactly, interprocess throughput can be increased since a task can guarantee a resource be made available for use by another (since it was allowed to complete).

Another choice which may need to be made available is the *run-to-completion* scenario, where a task will not be preempted until it voluntarily relinquishes the CPU. This is a solution which may provide some satisfaction to the executing task, though may upset the data entry staff. As such this system runs best on a multi processor system.

[3]The POSIX committee responsible for resolution of realtime issues is P1003.4A *realtime.*

[4]Some vendors have implemented realtime priorities through special values passed to nice(2).

Simulation systems require a different strategy again. The general requirement here is for a very precise scheduler, one that can switch in a task at a precise interval (usually down to a resolution of a couple of milliseconds) on a regular and frequent basis. The schedule of tasks to execute is given with respect to time. The schedule can be thought of as a frequency graph with a fixed repeating period. Because of this strict sequencing, these are known as *frequency based* or *synthetic period* schedulers.

The secret to any fast multitasking system is in its ability to context switch between tasks. There are many variables associated with performing a context switch. These include the possibility of having to swap out a process to make room for a new one, swap in a process to make it runnable or preempting a runaway task. Obviously optimising the methods mentioned here will improve context switch times and system throughput.

Task locking

Tasks may be spared the heartache of costly swapping activities by being locked in core using the *plock(2)* call provided under System V. Plock, for all its good, still does not permit the caller to lock its stack in memory, leaving it subject to attack from the pagedæmon under system load. This is an obvious oversight which must be fixed.

Kernel preemption

The most significant gains that may be made are through kernel preemption. Initially the kernel was a black box which having been given a job to do, only returns control once finished. Of course, if a signal was pending, control may not be returned since this is one of the times signals are delivered. The time taken to execute kernel calls is extremely variable with I/O between slow devices typically being the worst, providing delays of between a few milliseconds to in some cases many seconds.

If an event happens while an operator is retrieving medical history from mag tape, the delay can be chronic, resulting in significant loss of response time. Moreover, if many events happen during this interval, under current UNIX semantics, some of these events, and patients causing these events, may even be lost resulting in turmoil and lawsuits.

Kernel preemption provides a method of bypassing this carnage. By adding preemption points in various *slow* parts of the kernel, and providing sufficient locking algorithms to protect kernel data structures, it is possible to have a task switched partway through a kernel call, and have that call complete when the task gets rescheduled. The time savings are enormous. A typical system with preemption may be able to guarantee the longest unserviceable period being, say, 10 milliseconds. The actual time is not significant (although of course the smaller, the better), but the ability to know the worst case response time is.

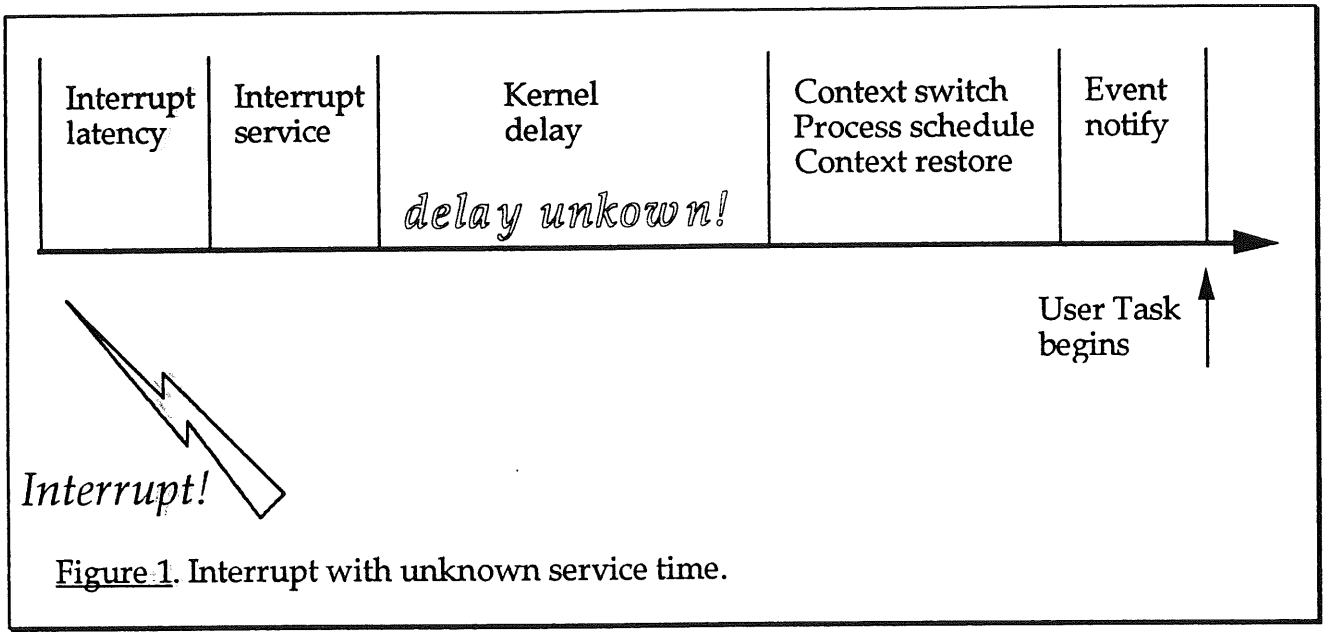


Figure 1. Interrupt with unknown service time.

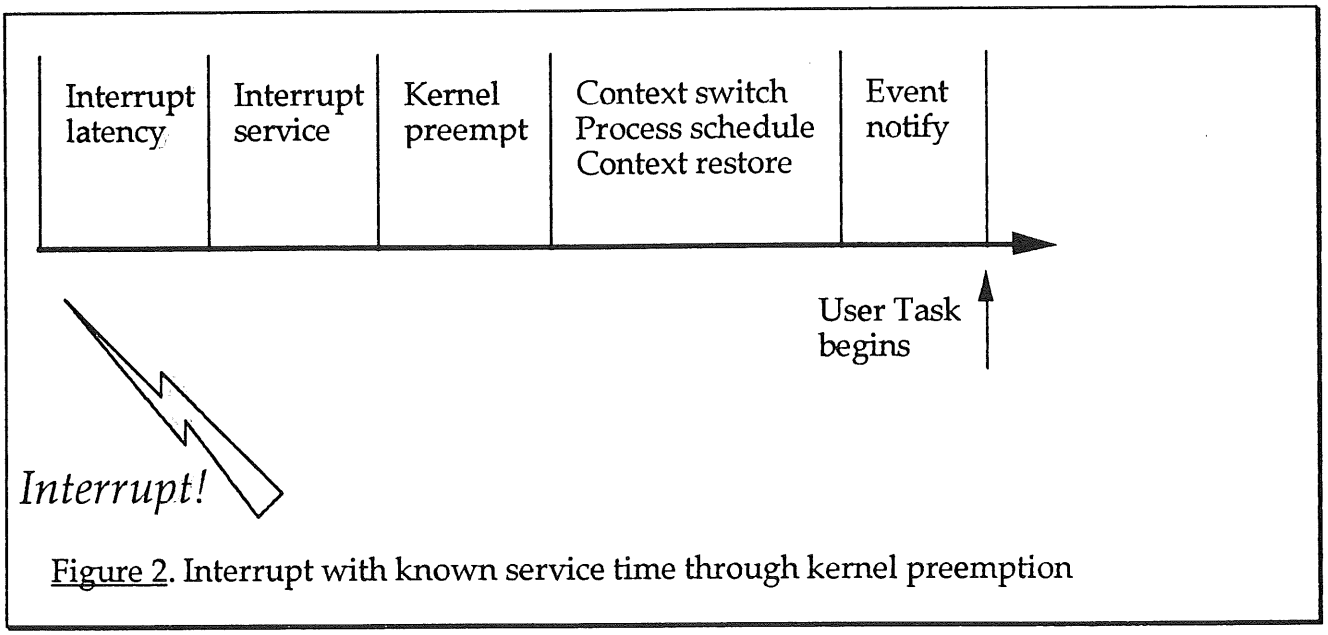


Figure 2. Interrupt with known service time through kernel preemption

Asynchronous interrupts

One of the most problematic calls ever devised under UNIX is the *signal(2)* interface. As Dennis Ritchie once wrote in a communication, the purpose of signals was to signify errors, they were not designed to be handled. This has become apparent with the amount of work that has gone into producing a *reliable signal* interface. Berkeley initially came up with the first reliable signal using *sigvec(2)*. This provided a mechanism to restart system calls interrupted by signals. The problem still remained that multiple occurrences of the same signal could be lost by not having anywhere to go, and finally, in a fit of depression, throwing themselves into the bit bucket.

To preempt discussion on modification to *signal(2)*, the following is an extract of a system call implemented on Concurrent's Real Time UNIX. The device is called an *Asynchronous System Trap* (AST). AST's are never lost, are queued, carry an associated priority for delivery, and allow one integer of data to be transferred. In all other ways they are identical to signals.

This allows a task to be interrupt driven and respond to events in a timely an prioritised manner. A priority threshhold may be defined which prevents AST's of priorities less than the defined maximum from being seen. Once the level is lowered however, they are seen and acted upon. By this means, a task monitoring the *nurse-call* buttons can produce a prioritised schedule of patients to visit during the busiest times.

Threading

Multiprocessors can readily make use of threaded applications. *Threads*[5], or *lightweight processes*, are tasks sharing a common address and (usually) data space. In this way the threads occupy no virtual or core memory, other than basic housekeeping space, but allow software implementation of various methods described in the earlier hardware section such as vectorisation, double buffering and pipelining.

Disk performance

Disks are typically the slowest link in a system. Realtime systems will be won and lost on the performance of their secondary storage media. When data acquisition rates exceed the capacity of available memory, throughput to disk is required.

All standard implementations of UNIX provide a tree structure in their file allocation algorithms. The first 10 or so data blocks are mapped directly within the inode. Pointer 11 points to a block which points to 128 more data blocks (first level indirection). Pointer 12 points to a block which points to 128 blocks which each point to 128 data blocks (second level indirection). Third level indirection is beyond the scope of the English language. A diagram is more useful.

[5]The working committee here is IEEE P1003.4A *pthreads* and P1003.14 *multiprocessing*.

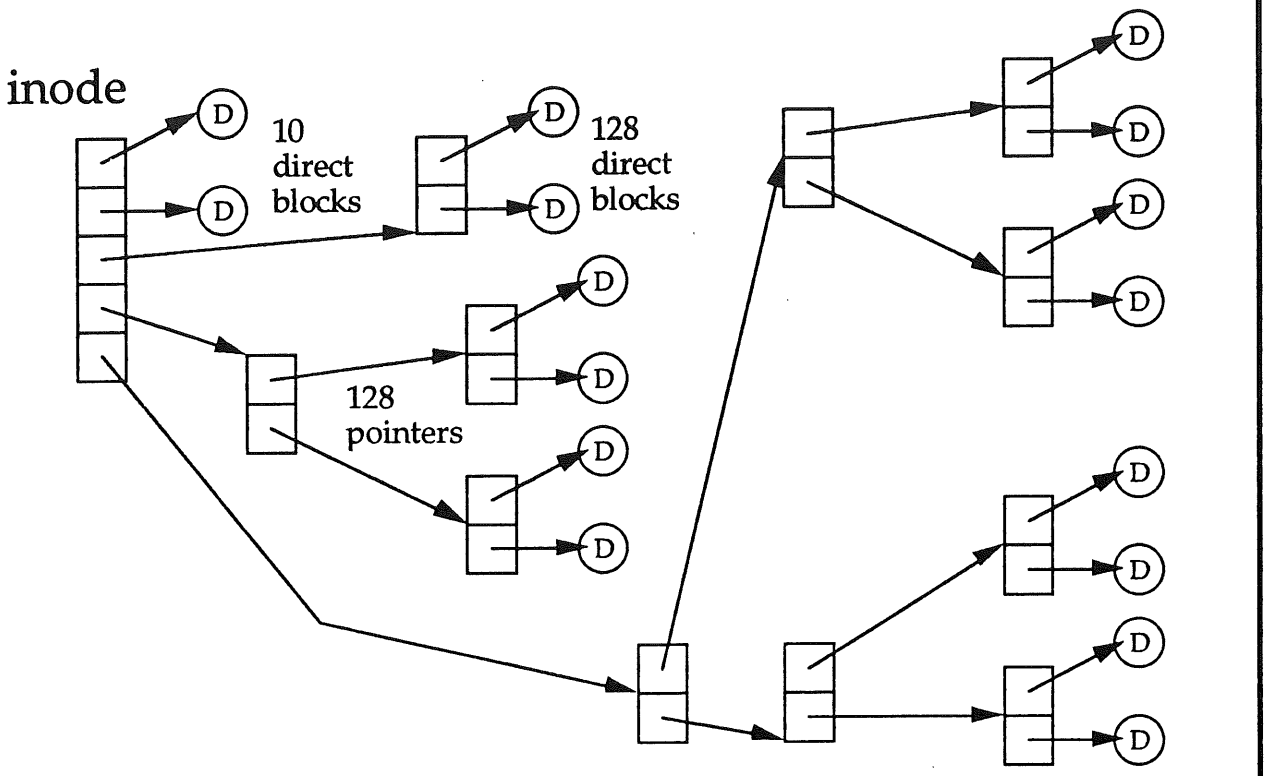


Figure 3. Layout of a typical UNIX file

Accessing a byte in blocks 1 to 10 require 2 disk reads. Any byte in the next 128 blocks requires 3 disk reads. After about 8 megabytes, the number of accesses required to read a single byte grows to 5. Although UNIX buffering usually does a good job of caching the information, the maximum delay is sometimes just too large an unpredictable to work with. The McKusick *fast file system*, found on most modern UNIX systems, is of no significant help here since dereferencing of index blocks still needs to take place at all indirection levels.

A solution is through *contiguous* files. This is where a file is preallocated as a consecutive number of blocks on disk. Accessing bytes in this type of file becomes a trivial exercise. By phasing reads and writes with the appropriate disk interleaves, it becomes possible to stream data to and from disk[6]. Furthermore, if data is transferred in multiples of disk block sizes, it is possible to perform *direct I/O* and bypass copying data to/from the system buffers. The flags argument to *open(2)* provides the obvious place to insert the new options of contiguous and direct I/O.

The implementation of contiguous files provides another interesting side effect. UNIX swap space is usually integrally tied into a disk at format time. This is required because the swap space must occupy contiguous space, previously not available under UNIX. By allowing UNIX to use contiguous files as swap, the need of preallocating swap space is alleviated. Adding swap space is as easy as creating a contiguous file.

---

[6]By streaming, I mean data can be continuously written without rotational delay.

A final point on UNIX I/O is the inability to either confirm or deny the existence of your precious data on disk. UNIX provides asynchronous writes which means that when a task writes information to disk, UNIX copies the write buffer and hands back control. At some later stage when there is less activity, this data is flushed to disk. While this allows tasks to continue quicker, it does not provide any guarantee of disk coherency should a power failure occur. Write(2) can best be summarised as follows[7]

> "I've taken note of your request, and rest assured that your file descriptor is OK. I've copied your data successfully and there is enough disk space. Later, when it's convenient for me, and if I am still alive, I'll try to put your data on the disk where it belongs. If I discover an error then I'll try to print something on the console, but I won't tell you about it (indeed, you may have terminated by then). If you, or any other process, tries to read this data before I've written it out, I'll give it to you from the buffer cache, so, if all goes well, you'll never be able to find out when and if I've completed your request. You may ask no further questions. Trust me. And thank me for my speedy reply."

The concept of a *synchronous write* must be implemented which does not return until data is secured to disk. Once again this can be implemented through the *open(2)* interface.

Summary

Most of the above mentioned strategies for dealing with realtime systems have been implemented in one or more systems, particularly Concurrent's Real Time UNIX. The bottom line still remains the same however, that until the POSIX committee ratify realtime guidelines, vendors will be unable to write truly portable applications.

It should be noted also, that although the aforementioned changes provide excellent solutions for realtime systems, they are trusted resources and as such can compromise the smooth running of a system if left in unskilled or malicious hands. Access to certain system resources must be controlled.

Acknowledgements

I would like to thank Peter Sneesby and John Hanna for their support without which I would not have got this paper together. Thank you also to staff at the fictional hospital for allowing me not to mention their names.

Bibliography

Bach, M.J. The Design of the UNIX™ Operating System
Rochkind, M.J. Advanced UNIX Programming
Henize, J. Understanding Real-Time UNIX®

[7] Extract is taken from *Advanced UNIX Programming*, by Marc Rochkind.

Making Real Use of a PC.

Frank Crawford

Australian Supercomputing Technology
(frank@atom.lhrl.oz.au)

1. Once Upon a Time ...

in a place far away (at least from Australia) some people wanted to play a computer game. Unfortunately there was nothing suitable so they found an unused machine and developed the ultimate games system called UNIX™ (:-)). The equivalent system today would be an IBM PC (of whatever brand), but there are very few versions of UNIX available for low end systems, such as PC/XT's. On the other hand there is an operating system which does work on these systems, it is MS-DOS.

2. MKS Toolkit

Looking at it another way one of the major advantages of UNIX is the utilities that are available. As most people would agree, the basic utilities supplied with MS-DOS leave something to be desired. But all is not lost, there are a number of packages available that can make you PC almost into a UNIX look-a-like. A number that will do this well are available from MKS. One of these packages is the MKS Toolkit which supplies over 150 different UNIX functions, from *asa* to *who* and *xargs*. The numbers are a bit misleading because the advertising people have counted a number of functions built into the Korn Shell (see more later).

2.1 Supported Functions

In terms of what is supported by MKS it seems to be a fairly complete set of System V (actually POSIX compliant) functions. Looking at the major functional groups, the Toolkit supports the editors *vi/ex*, *ed* and *sed*, the *grep* family, *awk*, *sort* and *spell*. There are also the standard file compression and archiving functions such as *compress*, *pack*, *tar*, *cpio* and *uuencode/uudecode*. As well as these, there is the Korn shell (*ksh*) and all its built in functions (things like *cd*, *alias* and *set* which area counted separately).

Aside from the programs, all the supporting files are also supplied, such as a spelling dictionary of nearly 100,000 words (for use by *spell* and *look*). In addition, they supply a number of manuals, which include a discription of all the functions (in traditional *man* style) and tutorials on *awk*, *Korn shell* and *vi*.

Finally, to give the *look and feel* of UNIX there are such programs as *init* (complete with *inittab*), *login* and *passwd*.

How is this done, you may ask? MKS have created an interface emulating UNIX system calls such as *stat(2)*. They have further written such programs as *glob* to provide consistent UNIX argument handling under MS-DOS's *command.com*. Obviously not all functions are compatible, so in these cases, they have implemented a logically equivalent function. For example, *ps* displays MS-DOS process and memory blocks and the process id is the MS-DOS *PSP* (program segment prefix).

2.2 Features

To the more interesting side of the MKS Toolkit, what sort of environment have they implemented. Personally, it is an excellent. There are many UNIX features that I wouldn't have imagined possible, but which are very useful. One of these is a simple job control, by this I mean that ^Z stops most MKS functions and *fg* allows it to be restarted. Obviously, this reduces the memory available for other jobs (as, like a TSR, the process remains in memory until recalled), but it is a vast improvement on starting another *command.com*.

One of the most sort after features (at least by UNIX users) is the ability to change option and path delimiters from '/' and '\' to '–' and '/'. This is accomplished by a program call *switch*, and works for most functions,

™ UNIX is registered trademark of AT&T in the USA and other countries.

including many MS-DOS internal functions (such as *dir*). Unfortunately, it doesn't work for all, *e.g.* MS-Windows expected to find a '\' in the path.

Another option to make the system more UNIX-like is the ability to replace the normal MS-DOS shell (*command.com*) by MKS's *init* which acts similar to the normal UNIX *init*, finally invoking *login* to handle the users login. *Init* is controlled by */etc/inittab* which can include TSR's and any other initialisation required, for example see Figure 1.

```
# Default /etc/inittab
# Put TSRs here as follows:
# t1;2;wait;c:/tsr/foo.com
fast;2;wait;dos/fastopen.exe c: d: e:
scsv;2;wait;lbin\setcge.com scrnsav on
tmpk;2;wait;lbin\timpark.com 5
sk;2;wait;dos/command.com /c c:\sk\skstart.bat
fr_c;2;wait;norton/fr.exe c:/save
fr_d;2;wait;norton/fr.exe d:/save
fr_e;2;wait;norton/fr.exe e:/save
rc;2;wait;bin/sh.exe etc/rc #Run commands
co;2;respawn;/bin/login.exe
```

Figure 1. Example */etc/inittab*

2.3 Differences from UNIX

Because of limitations with MS-DOS there are some things that the MKS Toolkit cannot emulate. There are two areas that this affects, process control/memory management and the filesystem.

2.3.1 Process Control/Memory Management As all MS-DOS users know it is not possible to run background jobs (the concept was not even known until *MS-Windows*). Because of this the *bg* function and *&* operator are not supported. Further, as all processes must fit in 640K of memory, there are many restrictions on job sizes (however, this is normal for any MS-DOS program). With the MKS Toolkit and its ability to stop jobs, which still occupy memory, this is sometimes a bit more of a problem.

2.3.2 Filesystem The biggest restrictions come from the MS-DOS filesystem and its restricted filenames (eight character plus three character extension). This means that such programs as *compress* cannot sensibly modify filenames, and so often they are restricted to writing to standard output and allow the user to redirect the output to a new filename.

Because of the directory structure, links are not supports and it is not possible to rearrange the directory structure, *i.e. mv* on a directory only allows renaming.

Also the information kept about files is not as extensive as UNIX, so programs such as *ls* are restricted in what they can report, *e.g.* all files are owned *uid* 0 (generally *dos* in MKS's password file). Finally, because of the use of ':' as a device designator, most of the system files use ';' as a separator.

2.4 Future Extensions

For the last two years MKS have been promising a upgrade which is yet to eventuate. One reason I suggest for the failure to deliver is that there is little they can improve on. The current version (3.1) seems to be both complete and reliable, I am yet to find any bugs in the implementation.

The only area that is lacking is communications. MKS would do well to include something like *UUCP*, or at least *kermit*.

One other possibility would be work on the memory usage, possibly make use of an Expanded or Extended Memory Manager. There is probably a need to do some reworking of *init* with MS-DOS 5.0, and the *loadhigh* command (*devicehigh* will still work from *config.sys*).

3. Other Products

Aside from the basic utilities there are many other products that make a UNIX system. Most of these are available for MS-DOS. Some of these are covered below.

3.1 Text Processing

One area that has long been well handled by UNIX systems is text processing, with *troff* and related packages. These facilities are readily available on PC, with such packages as *Eroff* (by Elan) and *SQPS* (by MKS). Both of these packages support AT&T's

Documenter's Workbench (*DWB*) and include drivers for HP LaserJets and Postscript printers.

There is very little to say about these products, except that they work. This paper was produced using the *eroff* package and then transfered to a UNIX system with a laser printer for a final proof.

3.2 Development

One of UNIX's strongest areas is in software development. Obviously in the area of C compilers there is a lot to choose from (most of which are similar to those available on UNIX systems), generally they are ANSI conformant, and so support the equivalent of many UNIX system calls, as well as other functions.

Products to support development such as *make* and *rcs* are widely avaiable (*e.g.* from MKS) all of which are derived from the UNIX originals. If there is any product you are currently using then there is probably a version available for MS-DOS.

4. Conclusion

In conclusion the MKS Toolkit and other MS-DOS packages can be used to make a PC look and feel like a UNIX system. But what is more, they allow you to still have access to all the other MS-DOS packages and they work with PC networks.

As an example of what you can do, this entire paper was written using *vi*, checked with *spell* and proofs generated with *troff*, all on a IBM PC/AT under MS-DOS 3.3, in effect no different programs to what I would use at on the UNIX system at work, and thus no new functions or procedures to learn.

5. Contact Details

MKS Toolkit, MKS RCS & MKS Make:

Mortice Kern Systems Inc
35 King Street North
Waterloo
Ontario
Canada N2J 2W9
Phone: +1 519 8844 2251
FAX: +1 519 884 8861
E-Mail: inquiry@mks.com

Australian contact:
Microway
P.O. Box 84
Mordialloc
Victoria 3195
Phone: +61 3 580 1333
FAX: +61 3 580 8995

The NeXT Computer - an Australian Perspective

Cameron Bromley

Codex Software Development Pty. Ltd.
(cdb@codex.oz.au)

The NeXT computer is now available in Australia. NeXT is running a developer camp in Melbourne on November 15th. The Australian pricing is good - a touch more than the US price converted to Australian dollars plus a bit more for shipping. More on the developer camp later.

Most AUUGN readers would know something about the NeXT. The original machine was announced some years ago now, and until recently was dormant. Things changed when NeXT announced the NeXTStation model in October last year. The NeXTStation is powered by a 68040 at 25Mhz, running Mach 2.1 and NeXTStep 2.0. Mach provides a 4.3BSD compatible operating system, with a few extensions - a faster, more consistent system of interprocess communication (ports), a larger virtual address space, memory mapped files and threads. NeXT have extensively used Free Software Foundation tools such as gcc and gdb (the excellent 'Gnu' C compiler and debugger), which are provided standard on machines with 400MB+ disks.

The 1.0 release was essentially a beta version of the product as it stands today. The 2.0 release is what I would consider the first commercial product. The interface itself is much more elegant, and all machines have at least 15MIPS and 8MB to power them. The philosophy of the machines has changed subtly from a rather esoteric workstation to a high-class personal computer. The slow optical drives are optional, rather than standard. The documentation is more user-oriented (although excellent technical documentation exists).

NeXTStep is a combination of the Workspace Manager, Interface Builder, the Application Kit and the Window Server. The Workspace Manager provides the user interface to the file system and Mach. The Window Server is essentially an implementation of Display PostScript - PostScript is sent to the server to draw images on the screen, and the Window Server will despatch events such as mouse clicks and keystrokes to your application. All drawing is done via PostScript routines; PostScript operators can be accessed as C functions. In addition, NeXT supplies a program named pswrap which will translate arbitrary PostScript into a C-callable function. The Display PostScript kernel is surprisingly fast, even without the NeXTDimension card, which provides an Intel i860 co-processor and 8MB of RAM to support full 32-bit colour and real-time displays. Included in the documentation is a chapter entitled 'Making Your Applications Fly', which deals with optimisation techniques and frankly discusses bottlenecks such as the window server and how to maximise the throughput.

Interface Builder is a very powerful application that has a two-fold purpose: It lets you graphically design a user interface for an application, and creates a programming environment for each project. To design an interface, you simply drag interface objects into the prototype application window. Interface Builder will create a '.nib' file, which contains the archived objects which make up your interface. This 'nib' file is loaded automatically at run-time, the objects de-archived and instatiated without the programmer having to worry about the details. Also created by Interface Builder is the primary Makefile (which contains targets for debugging, final builds and installation, among others). Interface Builder provides project management facilities, and will take care of the discrete components of an application. You can subclass any of the classes provided, Interface Builder will generate 'stub' source code for you and adds to the dependency list.

Interface Builder is extensible - example 'palettes' are provided which are loaded at runtime and provide interface objects such as simple bitmap 'paint' programs and 'smart' text fields which will validate their contents. Third parties are beginning to provide class libraries to add to the standard AppKit classes. Some of these are distributed via anonymous ftp, others commercially.

All NeXTStep applications use the Application Kit regardless of their purpose and complexity. The

buttons, sliders and and windows that are manipulated via Interface Builder are defined as Objective-C classes in the Application Kit. Application Kit classes include Button, Slider and Cell, as well as the more interesting LiveVideoView, Text and Application classes. The richness of the AppKit classes combined with the power of Objective-C and Interface Builder is, to me, the greatest attraction of the NeXT.

All the documentation is online and is accessed via the Digital Librarian application. The difference between the Librarian and every other online documentation facility I have seen is that it is useable, so much so that it is, to me, preferable to the printed documentation. The text is formatted, and the 92 dot-per-inch screen make it very readable. The Digital Librarian provides can index arbitrary directories. Simply dragging the graphical representation of the /usr/include directory (a folder) onto the Librarian application will provide text retrieval facilities for that branch of the file system. If an index is created, text expressions are matched virtually instantaneously and the results displayed.

The 'drag' facility provides some idea of the integration of the environment. Almost all application which have a representation of documents as part of the user interface provide dragging facilities, it is extremely easy to code the functionality required. The Workspace Manager will inform your application that the user has dragged documents over your window, and will also provide basic information such as the filenames, the document type, what applications recognise the document, etc.

Sometimes, the richness of the machine is it's biggest drawback - users and, especially, programmers of the machine are bewitched by the sheer volume of functionality and become NeXT propagandists. The drawback is that fanatics are viewed with a certain suspicion by everybody else. But it's very difficult not to become enthusiastic about this machine; so much functionality is provided with mostly no coding effort at all.

A good example is the Text class. You might need a text entry object as part of your interface. It's natural to drag a Text object from the Interface Builder application onto your window. That's the end of the story. From here, users can click on the text window, and start entering text via the keyboard. Or they can drag an existing document into the windows. Or the can paste via the system clipboard. TIFF and PostScript images can be embedded in the text. Sound objects which know how to play and record sound. Rulers with tab stops and margins. As many different fonts (of any size) that are on disk. If they are unsure of the spelling of a word, the users of your application can highlight the word and with a singe keystroke or menu selection define the word in the online Webster dictionary and thesaurus, which will suggest spelling alternatives if necessary - or provide a TIFF image if appropriate, which can be pasted back into the original text field. The entire contents of the text field, including sound and images, can be mailed to anybody via the same process. Or printed.

The really good part? - None of the actions above require any explicit coding.

If you, as a programmer, want to do something useful with the text, you ask the text field for, say, it's text. This is achieved via an Objective-C message:

```
[textThing getSubstring:buffer start:0 length:100];
```

This sends a 'getSubstring' message to the 'textThing' object, requesting it to place the first 100 bytes of the text it has into the memory location pointed to by 'buffer'.

The message syntax of Objective-C is reminiscent of Smalltalk. Objective-C is an extension of the C language; the Gnu compiler provided with the NeXT works with Objective-C, C and C++. Objective-C adds little to the syntax of the C language - a couple of new keywords and the [receiver message] expression. The language

The major difference between Objective-C and C++ is the dynamic typing - arbitrary messages can be sent to arbitrary objects at run-time. This feature is the underpinning of the ease of use and flexibility of

NeXTStep and the user interface. Static typing is available when needed, but is rarely used in practice.

As well as providing excellent programming facilities, the NeXT also provides front-end tools for user and network management, mail, printing, word-processing and faxing. All applications which print can also fax - the fax software is built into the standard 'print' dialog box. Unix can be 'hidden' via a Preferences switch; only /usr/bin in their Workspace Managers. Networking really is transparent, once the local domain server has been set up (a painless process). Newly attached NeXT computers will recognise the network, be assigned an Internet address (which can be overridden if necessary, and automatically mount public NFS volumes, gain access to printers, etc. After coming from a mixed Sun/Ms-Dos/Mac environment, I was very impressed by the ease of managing the entire setup.

It's difficult to convey the essence of the machine in writing - you really have to sit down in front of one for about half an hour to appreciate just how well built they are. Everything from the external casing to the icon design, the programmer API, the class libraries and the tools are well designed and elegant.

So, what does it all mean, in the end? The NeXT machines are neither the fastest nor the cheapest workstations to be found. Interface Builder is not the only product of its type. Other machines have Display PostScript and front-ends to Unix. But no other machine combines all this functionality and more into a package as economical, elegant and fun to use. Highly recommended.

The NeXT Developer Camp

Currently, the only way to purchase a NeXT machine and receive registered developer status is to attend the Developer Camp in Melbourne on November 18th. Those with registered developer status receive a 30% discount on NeXT hardware and receive excellent technical and developer support.

The idea is to 'seed' developers in Australia; developers will decide which machine they want before attending the camp, and take the machine home after the camp has finished.

Codex Software Development Pty. Ltd. are registered NeXT Developers and have no other affiliation with NeXT Inc. The camp itself and the training staff will be supplied by NeXT Inc.

This is a great chance for interested people to easily purchase a machine and gain developer status. The camp is limited to twenty people; bookings are on a first come, first served basis.

Codex Software Development Pty. Ltd., based in Melbourne, is acting as co-ordinator for the camp. Any queries, applications, etc should be directed to cdb@codex.oz.au or by phoning Cameron Bromley, Codex Software Development Pty. Ltd, voice (03) 696-2490; fax (03) 696-6757

On Semaphores for UNIX

Neil Dunstan

University of New England

Abstract

UNIX System V offers semaphores with significant enhancements compared to traditional Dijkstra semaphores. However, it is not always clear in what ways these additional features may be used. In fact, some writers on this topic are not in favour of their use at all. In this paper UNIX semaphores are described and compared to Dijkstra semaphores. Additional features are identified and practical uses are outlined. The discussion also identifies some inadequacies and suggests slight modifications to UNIX semaphore operations intended to overcome these difficulties.

1. Introduction

The semaphore [Dijkstra, 1968] is a commonly used synchronization mechanism. A semaphore is a non-negative integer with an associated queue of processes, which may be empty. The two indivisible operations which may be invoked on a semaphore are

P(s) wait on the queue until s can be decremented by 1

V(s) increase s by 1

Because of their simplicity and versatility semaphores have been included in a number of operating systems, including UNIX System V and OS/2.

2. UNIX Semaphores

The UNIX System V system calls semget and semop deal with arrays of semaphores rather than single semaphores. An array of a specified number of semaphores is created using semget. The returned identifier may then be used in calls to semop to perform operations on one or more semaphores in the array. Each operation is specified in a data structure of the form

```
struct sembuf{
        ushort sem_num;
        short sem_op;
        short sem_flg;
}
```

where sem_num identifiers the semaphore within the array, sem_op indicates the type of operation and sem_flg is used to qualify the operation to be performed.

The types of operations to be performed include

sem_op > 0	increase the semaphore value by sem_op
sem_op < 0	wait until the semaphore value is at least equal to |sem_op| then decrease it by |sem_op|

The operating system maintains a number of values for each semaphore. Among these are

semval	current semaphore value
semncnt	number of processes suspended due to an operation where sem_op < 0

3. Other Studies

Tutorial descriptions of UNIX semaphores are given in a number of books [Rochkind,1985; Bach,1986; Haviland and Salama,1987; Stevens,1990]. Rochkind and Haviland et al. suggest that there are too many features, creating a complexity that is not conducive to reliable concurrent programming. Rochkind in particular advocates a simplified interface to the UNIX system calls that reduce their features to that of Dijkstra's P and V operations. While this provides a useful implementation of the common notion of semaphores it seems unnecessarily cautious to avoid all the additional capabilities of UNIX semaphores. Stevens does make use of some of these additional features but shows that the inability to provide an initial value to a semaphore at the time of its creation can cause problems that require a complicated and not altogether adequate solution. In another paper [Dunstan and Fris,1991] UNIX semaphores are evaluated with respect to their ability to support different process scheduling strategies.

4. Making use of UNIX Semaphores

Several ways in which UNIX semaphores provide capabilities that are not provided by Dijkstra semaphores are discussed in this section. In each case, practical applications are given.

4.1. Arrays of Operations

It is possible to build an array of operations to be performed indivisibly on a semaphore array. A process invoking an array of operations is suspended until all can be achieved. If one operation, for example decreasing a semaphore value, cannot be done, then none are done. When all operations can be done, they are done indivisibly.

Semaphore solutions to the Dining Philosophers problem [Dijkstra, 1968] typically use serially acquired forks (represented by semaphores with initial values of 1) and require extra code to avoid the possibility of deadlock. That is, each philosopher acquires exactly one fork and is left waiting for the other. The problem solution using UNIX semaphores has each philosopher acquire (or wait for) both appropriate forks indivisibly using an array of operations.

4.2. SEM_UNDO flag

Setting sem_flg equal to SEM_UNDO qualifies an operation so that the semaphore value is adjusted appropriately if the calling process exits.

This facility has obvious use in critical section problems, that is, when a number of processes must execute a code section in a mutually exclusive fashion. Dijkstra semaphore operations can be used to surround the section. For example ...

```
P( mutex )

/* critical section */

V( mutex )
```

where mutex is initially 1, allowing one process at a time into the critical section. Should a process exit within the critical section before calling V(mutex), the semaphore value is not restored and all other processes are deadlocked. A UNIX semaphore solution, using sem_op = -1 with sem_flg = SEM_UNDO as the P(mutex) operation, would again avoid deadlock with minimal additional code. The V(mutex) operation must be replaced by sem_op = 1 and sem_flg = SEM_UNDO.

4.3. IPC_NOWAIT flag

Setting sem_flg equal to NO_WAIT qualifies an operation that may result in a process being suspended. The effect is that, if possible, the operation is done immediately. If it cannot be done immediately then the operation is aborted and the process is not suspended.

When sem_op = -1 and sem_flg = IPC_NOWAIT, the operation is the equivalent of the tryP operation included in the set of synchronization mechanisms for the MONADS-PC system [Dunstan, 1989]. It allows a process to secure a resource (protected by a semaphore) if it is immediately available, or carry out some other useful work if not.

4.4. |sem_op| > 1

Operations with the absolute value of sem_op > 1 shows a marked departure from Dijkstra semaphores. A practical use is given in [Dunstan, 1991] for a semaphore implementation of monitor priority conditions.

The semaphore implementation of monitors and conditions is reasonably straightforward [Hoare, 1974; Peterson and Silberschatz, 1985] but prioritized queuing of conditions greatly complicates the algorithms where Dijkstra semaphores are used. However, a priority condition queue can be conveniently represented using a single UNIX semaphore upon which processes are suspended using sem_op as their priority level. Signalling a condition is done by progressively incrementing the semaphore until a process is released (the one with the lowest priority level).

Another application (discussed in greater detail in [Dunstan and Fris, 1991]) is an

implementation of the eventcount operations Advance and Await [Reed and Kanodia, 1979]. Advance(event) increases the value of an event by 1 and Await(event, level) causes the calling process to remain suspended at least until the value of the event has reach the stated level. An eventcount can be represented by a UNIX semaphore where the Advance operation simply uses sem_op = 1. The Await operation uses sem_op = -level and then immediately restores the eventcount value by a corresponding operation, that is, sem_op = level. In this way, the net effect is that the eventcount continues to rise, allowing processes waiting for higher levels to be released appropriately.

5. Recommendations

While the features of UNIX semaphores provide convenient solutions to some problems they might easily be more efficient.

The implementation of monitor priority condition queues involves potentially many system calls if there are many priority levels. This could be avoided by employing a new flag SEM_RELEASE (to be assigned to sem_flg) whose meaning is to qualify an operation where sem_op > 0 so that the value of the semaphore is directly raised to the minimum required to release a process. With this flag, only a single system call is required to release a process suspended on a semaphore used for prioritized queuing.

The implementation of the eventcount operation Await uses two system calls. The first waits until the semaphore reaches a given level and then decrements it to 0. The second system call then restores the value. Clearly, the second system call is not necessary if a process could wait for a given value to be reached without changing it. Another new flag SEM_NOCHANGE is proposed which qualifies an operation where sem_op < 0 so that the process waits as usual, but does not change the value of the semaphore.

The meaning of the proposed flags are summarized below

```
SEM_NOCHANGE  if sem_op < 0 then
                      wait until semval >= |sem_op|
              else
                      no additional effect

SEM_RELEASE   if (sem_op > 0) and (semncnt > 0) then
                      semval is assigned the minimum of n
              else
                      no additional effect
```

(where n is the set of values by which processes are waiting to decrement semval)

6. Conclusions

UNIX System V semaphores offer many more features than traditional Dijkstra semaphores. Rather than being unwanted and irrelevant, these additional features have real and practical applications. Despite the substantial code and data structures associated with the semaphore

system calls, there are some inadequacies that might easily be remedied. Additional flags have been proposed for this purpose which would not significantly add to the code and data structures involved.

REFERENCES

M. J. Bach, *Design of the UNIX Operating System*, Prentice-Hall, Englewood Cliffs, N. J., 1986

E. W. Dijkstra, "Cooperating Sequential Processes", in F. Genuys (ed.), *Programming Languages*, Academic Press, New York, 1968

N. Dunstan, *Concurrent Programming in LEIBNIZ on the MONADS-PC System*, M. Comp. Sci. Thesis, University of Newcastle, 1989

N. Dunstan, "Building Monitors with UNIX and C", University of New England, Dept. of Maths., Stats. and Comp. Sci., *Technical Report 91-36*, 1991, To be published in ACM SIGCSE Bulletin

N. Dunstan and I. Fris, "Process Scheduling and UNIX Semaphores", University of New England, Dept. of Maths., Stats. and Comp. Sci., *Technical Report 91-37*, 1991

K. Haviland and B. Salama, *UNIX System Programming*, Addison-Wesley, Great Britain, 1987

C. A. R. Hoare, "Monitors: An Operating System Structuring Concept", *Communications of the ACM*, Vol. 17, No. 10, Oct. 1974.

J. L. Peterson and A. Silberschatz, *Operating System Concepts*, 2nd Edition, Addison-Wesley, USA, 1985

D. P. Reed and R. K. Kanodia, "Synchronization with Eventcounts and Sequencers", *Communications of the ACM*, Vol. 22, No. 2, Feb. 1979

M. J. Rochkind, *Advanced UNIX Programming*, Prentice-Hall, Englewood Cliffs, N. J., 1985

R. Stevens, *UNIX Network Programming*, Prentice-Hall, Englewood Cliffs, N. J., 1990

An SNMP Stereo System:
Musical Networks, Australian Style

by Simon Hackett

At INTEROP 90, I had the pleasure of demonstrating a device which brings something of a new twist to the use of the *Simple Network Management Protocol* (SNMP) on a TCP/IP network.

SNMP

SNMP is designed to allow vendor-independent monitoring and management of objects on a TCP/IP network. It is conventionally used to monitor and manage devices such as IP routers and bridges, and compute hosts on an IP network. It is in essence a protocol which allows an IP node to provide access, over the network, to a tree-structure of variables. Variables can be read-only or read/write. The ability to write values into variables is controlled by a fairly rudimentary scheme of "community" strings—essentially a plain-text password.

There is a standard tree of variables, a so-called *Management Information Base* (MIB), which each vendor of SNMP products should support. These allow the retrieval of information about the IP node—information such as IP network numbers, routing tables, and packet input, output and error counts.

An unusual application

New tree structures (MIBs) can be defined to suit other applications. To prove the point, I demonstrated an unusual application—a networked stereo system, using SNMP protocols to provide full monitoring and control of the system. The stereo system we demonstrated was a Pioneer Tuner/Amp (with a cute motor-driven volume knob), a Pioneer PD-M910 six-disc CD player, and a set of Klipsch speakers. We operated the stereo system for the interest of visitors to the TGV booth, using X-windows front end software plus some simple command-line based SNMP tools.

Hardware

Connecting this system to the network was a little "magic box." Based on a Motorola 68000 processor, this device performs real-time monitoring and control of the stereo system, and also runs an IP kernel, with ICMP, UDP, and an SNMP agent which implements our audio-visual MIB. It talks IP protocols to the world using serial line IP (SLIP). Interfacing the to stereo system is achieved using two connections. The device listens to the signal coming from the the "Digital Output" jack on the CD player, and generates a control signal which connects to the Pioneer-standard "remote control input" jack on the CD player. The "remote control output" on the CD player is daisy-chained to "remote control input" on the tuner/amplifier, so the device can control both units. (See Figure 1).

Software

The software in the box was written almost entirely in **C**, using Sun workstations. Much of it was written during a period of *intense* activity at TGV Inc in Santa Cruz, California during August of 1990, when I worked with several others to finish the software in the device, adding the IP and SNMP support to the control software I had previously written.

The control software uses some careful timing to imitate the remote control signals used by Pioneer's stereo components. Thus, by being plugged into the "remote control in" jack on the CD player, any CD player or Tuner/Amp function can be initiated by the controller. The digital output signal from the CD player contains both the digital audio samples for the music being played, and a stream of status information.

This information is decoded in real-time using interrupt driven routines to provide the system with continuous monitoring of the position of the CD player in any music selection played. The "table of contents" information from each CD is also read using this interface.

Using the system

The device implements a 100 entry play queue for the CD player, much like a jukebox. It processes this queue, playing selections, and allows any connected network node to monitor this queue, add selections to it, and do other monitoring and control of the stereo system (tuner/CD selection, amplifier volume adjustment, fade down, fade up, etc). SNMP operations allow full management of the queue, including retrieval of queue elements and insertion, deletion and replacement of entries anywhere in the queue.

For INTEROP, we wrote some demonstration applications to show off the unit. A set of X11 tools show the status of the system (including the currently playing disc number, the playing time of the current selection in minutes and seconds, the amplifier volume, tuner status, etc). Recognizable buttons are provided in the window to provide the play, pause, stop, eject and other required functions. (See Figure 2).

Another window displays a set of disc titles from a database of available discs. Clicking on a disc title pops up a window listing all the tracks on that disc, and also all the tracks on the other discs in that "six pack." Clicking on track titles in the window causes the appropriate SNMP requests over the network to the controller, which appends the selection to the play queue.

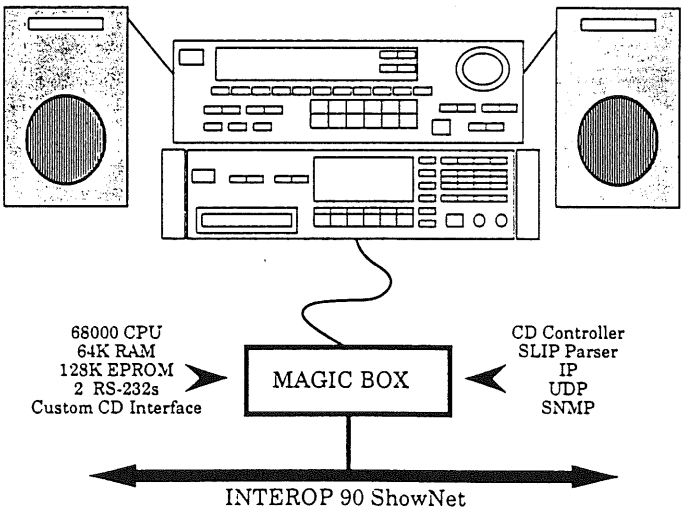


Figure 1: Hardware configuration

SNMP control issues

With Marshall Rose of PSI Inc and Stuart Vance of TGV Inc, I developed an "audio-visual" MIB, providing a tree structure which describes the functions and information available from a range of stereo components. This MIB is quite substantial, since it describes all the functions of the six disc CD player, and most functions of the Tuner/amp unit. It contains sections for most other sorts of audio/visual components, which we will "fill out" as needed.

Because the SNMP protocol is general, any software capable of issuing SNMP *set* and *get* requests is capable of accessing the stereo system, by using our custom MIB to find the identity of the required variables. This includes line-based SNMP libraries available from several sources, and SNMP network management station software.

An SNMP Stereo System *(continued)*

Control and monitoring of the stereo is effected with sub-trees of SNMP variables describing each function. For instance, one sub-tree contains current status information for the CD player (current disc, current track, time into track, time into disc etc). This sub-tree can be "walked" using SNMP's *powerful get-next* operator, to display the player's status. The values returned are real-time information from the CD player.

Another sub-tree contains information describing each disc in the six pack. If less than six discs are loaded, the sub-tree simply shrinks as appropriate. For each disc, a further sub-tree gives an identity number for that disc, the number of tracks on the disc, and the duration of each track.

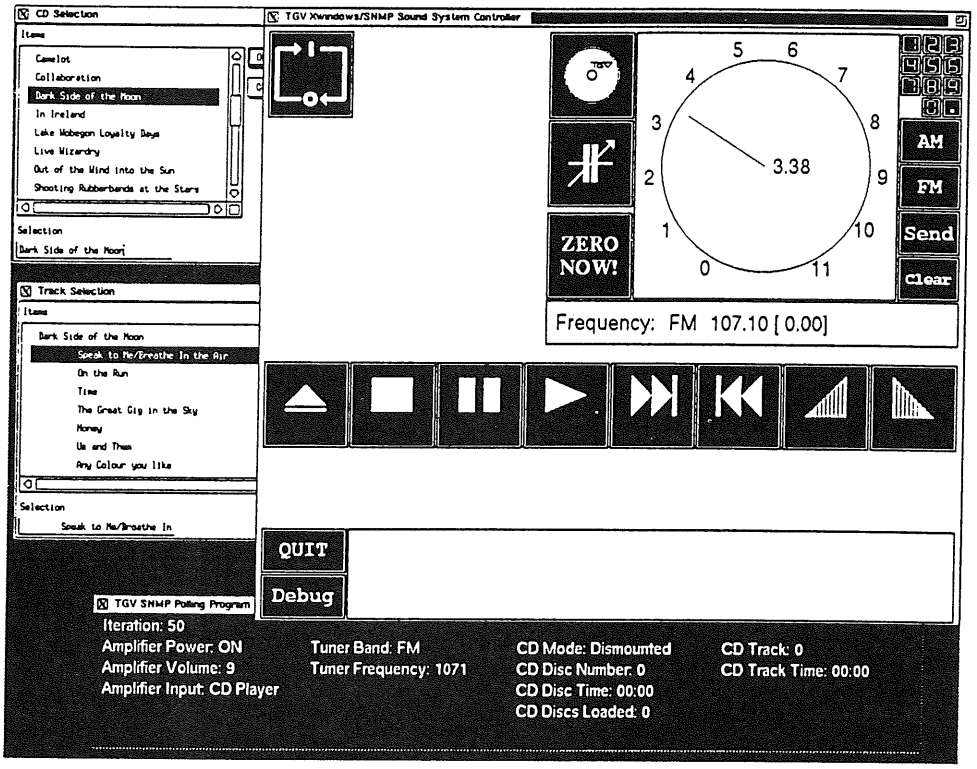


Figure 2: The X Tools CD player control window

Yet another sub-tree is responsible for the queue of tracks waiting to be played. For each entry in the queue, a part of this tree describes the attributes of that selection. These are: disc ID, track number ("0" implies the whole disc), starting and ending times, plus track repeat and requeue counts. Each entry in the queue is assigned a unique 32 bit ID by the software for later reference, and this ID also appears as an SNMP variable in each queue entry. Walking this sub-tree provides a list of the tracks waiting to be played.

To instruct the system to queue a selection, a set of SNMP variables is written in as a "request block," and then a function code is written into a variable to tell the system to queue the entry. Functions have been implemented to allow queue entry insertion, deletion, replacement, etc. Other system settings are accessible via SNMP. For instance, one variable is the amplifier volume setting. This can be changed with an SNMP *set* request to change the volume, and read using an SNMP *get* to check the current volume level.

Wide area access

Since the controller is a functional IP node, it can be accessed anywhere on the internet it is connected to. Indeed, I have gained some amusement from altering the volume setting on a system at TGV in Santa Cruz, CA, from Adelaide, South Australia over the Internet. This is quite a good computer-hackers' party trick.

Conclusion and futures

At INTEROP 90, we demonstrated that SNMP can control anything you want to control. Indeed, the backup controller device was used to operate the *second* Internet Toaster to make an appearance at INTEROP 90; the first Internet Toaster was demonstrated at INTEROP by John Romkey. We also found out that the majority of networking people at INTEROP 90 seemed to prefer listening to Pink Floyd and Monty Python (almost to the exclusion of anything else, it seemed...)

I hope to demonstrate the next generation of this device at INTEROP 91. It will be using Ethernet instead of a serial line for the IP connection, and instead of just monitoring the stereo system, I intend to push the digital audio data from the CD player down the network and pop it out of another black box somewhere else on the show floor! I should also have some other interesting applications of this technology ready to demonstrate at the show.

Acknowledgements

This enterprise would not have happened without the generosity and encouragement of TGV, Inc in Santa Cruz, California. I need to thank Ken Adelman and David Kashtan of TGV for their support (including flying me over to make this happen), John McMahon of TGV for writing the X-windows code. John Romkey and Karl Auerbach of Epilogue Technology assisted us and kindly allowed us to use their SNMP agent code as the core of our SNMP implementation. Most importantly, I am very grateful to Stuart Vance of TGV for his friendship and encouragement, and for his contribution of a great deal of his time and effort to help make this happen.

This paper has been reprinted with permission from ConneXions, Volume 5, No. 4, April 1991.

ConneXions--The Interoperability Report is published monthly by:
Interop, Inc.
480 San Antonio Road, Suite 100
Mountain View, CA 94040
USA
Phone: (415) 941-3399 FAX: (415) 949-1779
Toll-free (in USA): 1-800-INTEROP

Free sample issue and complete list of back issues available upon request.

An Update on UNIX–Related Standards Activities

Jeffrey S. Haemer

Report Editor, USENIX Standards Watchdog Committee

1003.14: Multiprocessing

Bill Cox <bill@attunix.att.com> reports on October 15–19, 1990 meeting in Seattle, Washington:

P1003.14 (Multiprocessing) Summary — Seattle

Dot14 is working toward a first draft of its document, a multiprocessing Platform Environment Profile (PEP). (The terminology has changed; this is more or less what used to be called an Application Environment Profile.)

Bob Knighten (Chair) and Bill Cox (I'm the Secretary) are going to edit the most final UI Multiprocessing Working Group (MPWG) report and replace the copy apparently sent in error by UI. The working group adopted the to-be-delivered version as a base document for the standard.

The UI MPWG document does not address coherence, sequentiality, or the management of a multiprocessor, all of which are important areas to the Dot14 working group. Non-Uniform Memory Architecture (NUMA) issues are also a consideration; the UI MPWG report deals primarily with Uniform Memory Architectures.

The group is starting to produce a coordination ballot for P1003.4A Pthreads, which was released for ballot by the Seattle meeting. Since the Pthreads work has now wound down (the draft standard is now in the hands of the technical reviewers, not the working group), many of the partisans that were attending Pthreads will start attending Dot14, and therefore will have a role in shaping the coordination ballot on their own handiwork. There will be a separate effort to create a Common Reference Ballot, similar to the one produced on Dot4, but due to the changing composition of Dot14 that work will take place informally and outside of Dot14.

Note that a coordination ballot from another POSIX group can only be a "NO" ballot, and presumably draws significant attention (as do all of the institutional ballots).

The work items for the New Orleans meeting are:

1. Prepare coordination ballot for Pthreads.
2. Review draft 2 of Dot14 PEP.
3. Review state of the "TIMS" PEP.
4. Coordinate with the Dot13 AEP.
5. Continue working on modifications to and issues with Base Standards.
6. Continue to coordinate with other working groups.

The current list of coordination issues with other POSIX working groups is:

Dot2

- Parallel utilities, including *make*, *fsck*, *grep*, and *find*.
- File system directory tree walk.
- How to specify parallelism (command line flags, environment variables,...).
- Specification of the effects of parallelism. Does a "parallel" version have to be faster if processors are added?
- To what extent can parallelism change the effect, e.g., output order?

Dot1 and Dot4

- Resource reservation.
- Processor binding and scheduling.
- The *plock* function.
- Shared memory interfaces.

Dot4A (Pthreads)

- Microtasking models (finer than Pthreads).
- Resource query and brokering.
- NUMA configuration and control.
- Synchronization primitives, e.g., barriers, that are not in the Pthreads draft.
- Re-entrant interfaces and functions.
- Memory model (the Pthreads model is overly restrictive in a multiprocessor).
- Scheduling (this seems to have cleared up with recent changes in Pthreads).

Dot7

- Resource query (see also Dot4A list).
- Statistics gathering and display.

Dot8

- Resource brokering (again).

X3J16: C++

Mike Vilot <mjv@objects.mv.com> reports on the November, 1990 meeting in Cupertino, California:

Current Status

The ANSI X3J16 committee closed out their first year of discussions by completing the definition of the C++ language. At the November meeting, they agreed to incorporate the terminating exception proposal (the text from Chapter 15 of *The Annotated Reference Manual*, minus the annotations, plus some minor clarifications). C++ vendors can now regard templates and exceptions as officially part of the language, and provide their users an opportunity to work with this feature.

We saw some progress on the review of language ambiguities and inconsistencies, and are beginning to get some idea of how difficult it will be to ANSI-fy the document. We also saw some specific proposals on library contents (the most substantial suggestion to date has been a simplified version of the *iostreams* library).

November meeting

Hewlett-Packard hosted the Cupertino meeting. The week's major activities focused on settling the debate surrounding the design of the exception-handling facilities in C++. There was also a rather long discussion of internationalizing the development of the C++ standard.

X3J16's sub-groups focus on the key topics listed in the goals statement developed at the March meeting. They worked by electronic mail between meetings, and reported their progress.

International Concerns

Steve Carter of Bellcore presented the major international concerns.

Steve explained the differences between "Type D" (domestic first) and "Type I" (international) procedures in preparing an international standard. His group suggests converting to a "Type I" process, which results in simultaneous review and standardization at the national and international levels.

One aspect of this conversion that caused the most discussion was the ISO requirement to provide a detailed explanation of the "incompatibilities" between C and C++. As Bjarne Stroustrup observed, that term is loaded and causes a shift in the group's emphasis to defining the "not C" elements of C++ and defending why they are there. He also pointed out that X3J11 was not required to go through this sort of exercise.

The committee formally moved and tabled a decision to make the conversion. This puts the group on record as having considered the idea, but expresses their reservations about the various implications of such a decision.

We also discussed the alternative to trigraphs proposed by Keld Simonsen and Bjarne Stroustrup. There is some confusion among X3J16 members regarding the extent of systems supporting ISO 646 and the Latin-1 character set. X3J16 will investigate the issue in more detail at the March meeting.

There was also some discussion of the normative addendum to ISO C proposed by the Japanese. Since this will affect the goal of maintaining compatibility between C++ and C, this issue will bear watching.

Editorial

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

The most significant change since the July meeting was a clarification in Chapter 11 regarding protected derivation. The language described as of the 2.1 release of *cfront* slightly generalized the language to allow all three access specifiers for derived classes. Jon described the changes to the language specification as clarifying this point and simply explaining the semantics involved.

An important development was the lack of progress on the Rationale document. John Dlugosz had volunteered to edit this document, and has not been heard from since the July meeting. He was not present at the November meeting.

Formal Syntax

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

He continued to lobby X3J16 to accept his copyrighted *yacc* grammar for C++ as the formal definition of the language. He produced little evidence that the group worked on reviewing either relevant formal methods or the applicability of those methods to the development of the standard.

Mark Langley of Microsoft presented ten examples that illustrated what he considered problems in the language. Bjarne Stroustrup and others pointed out that most of the issues are answered by the text of the language specification. The committee expressed a general dissatisfaction with the group's process of random fault-finding, and strongly urged the group to present specific proposals for language changes and/or clarifications in the language specification.

Tom Penello of MetaWare presented an analysis of the impact of the template syntax on the LR(1) properties of the grammar, using a grammar analysis tool of his own design. While the committee appreciated the value of automated tools to pinpoint problems in the grammar, the consensus of the committee was that an entirely grammar-based specification of the language was unrealistic — even if the grammar could precisely and unambiguously specify the language, the result would likely be too complex to be useful to either implementors or users.

Core Language

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

The group came up with a list of almost 80 issues that need further clarification. They also adopted a policy of not considering purely aesthetic changes.

An example of the kind of issue they are discussing is a clarification of the rules concerning creation and destruction of compiler-generated temporaries. These rules need to be clarified, because construction and destruction of temporary objects of user-defined types can be potentially expensive. It is also important to avoid placing too many limits on an implementation, precisely because vendors would like to be able to optimize away unneeded temporaries.

Environment

John Vasta of HP presented the work of the Environment group.

Much of their discussion revolved around precisely specifying the interaction with the environment during linking and execution. They devoted considerable effort to the "one definition rule," which requires exactly one copy of each function and object. There was also some discussion of the semantics of static initialization, especially in a mixed C++/non-C++ program.

The emerging consensus of the committee seems to be against requiring all implementations to provide "hooks" to underlying details of implementation. These include non-C++ access to static constructors and destructors, a standard "de-mangling" function, and a required type-safe linkage encoding scheme.

Jerry Schwarz of Kubota Pacific (formerly called Stardent), presented some of the group's discussion on the topic of mixed C and C++ environments. Some of the issues include type equivalence between C and C++, name aliases between the languages, and details of function invocation in a mixed-language program. This topic overlaps both the C compatibility issues and the C library topics.

C Compatibility

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

He presented a list of definitions from the C standard that should be incorporated in the C++ standard, as document X3J16/90–0088. The committee accepted his suggestion that Jon Shopiro incorporate the definitions in the next revision. The challenge will be to review the document to ensure consistent and appropriate use of terms throughout.

Libraries

I presented the Library group's work.

The main results so far are proposals for standard definitions of three library areas: lan-

guage support, iostreams, and strings. Documents X3J16/90-0077, -0078, and -0079 contain the details of the proposals.

Alan Beale of SAS provided many comments on the streams proposal. Some aspects of these classes contained UNIX dependencies, and were difficult or impossible to implement in other environments. Jerry Schwarz also suggested incorporating exceptions in a way that preserves current practice — the choice of whether to throw exceptions or set a state variable could be under programmer control.

Steve Clamage of TauMetric presented a summary of the issues surrounding support for the ANSI C library within the C++ standard (document X3J16/90-0105). Progress on this issue has been slow, because each person who has volunteered to address the issue at one meeting has left X3J16 by the next meeting.

Aron Insinga of DEC presented his proposal for a standard string class. There are many members of the committee who have implemented successful string classes, so there was much useful comment on Aron's proposal. Many of the suggestions exploited advanced features of C++, such as templates. Bruce Eckel of Revolution2 suggested that at least one of the standard string classes be kept simple enough to be useful as a tutorial example for new C++ programmers.

Jim Howard of Mentor Graphics provided a string class he developed. Notable aspects of Jim's library included support for Japanese characters and the success the library has enjoyed with clients around the Pacific Rim. This library could provide valuable input for addressing the international concerns about multinational character handling. It may be possible to avoid a language extension by providing the necessary support in the library.

Language Extensions

Bjarne Stroustrup of AT&T presented the work of the Extensions group, which was by far the most active.

The key discussion of the week was of course the design of exceptions for C++. Since the March meeting, the issue had been whether to incorporate just the terminating model, or also incorporate resumable exceptions. At the July meeting, Martin O'Riordan of Microsoft had asked for time to do further research on the topic. He proposed adding constructs to allow programmers to specify resuming and non-resuming exceptions and handlers.

The key point in the discussion came during a discussion of exceptions by Jim Mitchell of Sun Microsystems. His long experience designing languages with exceptions, using such languages to build large systems, and efforts to document and teach the use of resumable exceptions, all helped to clarify the discussion. The final vote was 30–4 in favor of accepting the text of Chapter 15.

The next substantial language extension will be consideration of a standard way to provide run-time access to type information. Mark Linton of Silicon Graphics outlined one proposal in his Dossier concept (see: "Runtime Access to Type Information in C++" at the 1990 USENIX C++ Conference, San Francisco, April 1990).

Next events

Now that the major design decisions are completed, the task of standardizing C++ will get down to details. We can expect that any changes to C++ will be relatively minor — certainly less than, say, the introduction of function prototypes in ANSI C.

The target date for delivering a draft C++ standard will be affected by the involvement in ISO. One informal estimate is an additional year to complete the standardization process (which would mean a simultaneous national and international review in 1993). The next three X3J16 1991 meetings (and their hosts) will be:

- March 11–15, Nashua, NH (Digital)
- June 17–21, Lund, Sweden (Lund Institute of Technology)
- November 11–15, Toronto, Canada (IBM)

Texas Instruments will host the March 1992 meeting in Austin, TX. 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@hda@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

Book Review
PROGRAMMING IN PERL
by Larry Wall and Randal Schwartz

(O'Reilly & Associates, 1990, ISBN 0-937175-64-1, $24.95, 454 pages)

Reviewed by Tom Christiansen and Rob Kolstad

The O'Reilly folks have released another Nutshell Handbook, this time on the *perl* programming language. *Perl* is the most significant general-purpose tool to hit the UNIX world in years, filling a niche between shell programming and C programming. We hope and believe it will soon become a standard utility (like *awk* and *sed*).

If you haven't used *perl*, it's a lot like C with *awk*, *sed*, *grep*, shell programming, and just about everything else included. The magnitude and quality of the synergy is surprising. Problems difficult or impractical to code in shell scripts are often much easier to write and faster to run when transcribed into *perl*. The language has been described as a shell for C programmers, a BASIC for UNIX, and even an APL on LSD. Because *perl* is interpreted, incremental development and testing is simple. Its powerful constructs combine with efficient execution to yield a pleasant programming environment.

Another brainchild of Larry Wall (who gave the world *patch* and *rn*), *perl* has enjoyed ever increasing popularity in the UNIX community since its initial public release about four years ago. In particular, system administrators have found it handy, since they're the ones most often tasked with writing or maintaining convoluted shell scripts. Some test and manufacturing groups at computer vendors are now considering using *perl* as their one-and-only language for test engineers.

The *perl* language is available without cost from various FTP sites on the Internet. It's also on the GNU tape distributed by the Free Software Foundation, although it isn't GNUware *per se*. Friends of *perl* distribute it on request to their less-connected neighbors. While few formal corporate entities yet back *perl*, its quality is typical of Larry Wall productions. Support comes quickly from the comp.lang.perl newsgroup (or electronic mail to any of the USENET *perl* gurus). Most questions regarding *perl* resemble "how do I do this" rather than "this appears to be a bug".

Until now, *perl* programmers have had to make do with the meandering, 75-page *man* 'page', a bit intimidating for people just trying to get a handle on the language. For this reason, if no other, this book by Larry Wall and Randal Schwartz is welcome.

The book is hefty (1.1 inches thick; 6x9 inch format) with 450 pages of densely packed material. It is organized into seven chapters, starting with 'An Overview of Perl,' which explains the basic concepts of *perl* in a fairly informal fashion. The phrase 'informal' is probably an understatement; this is definitely not a dry and stuffy technical book. Like the man page (once described as more of an editorial than a reference guide), the book tries to entertain you as it teaches. Whether it actually succeeds at this depends on your particular sense of humor.

The second chapter, 'Practical Programming,' is also easy to read. The authors show the kinds of problems *perl* is good at solving and ways to use *perl* to solve them. It begins with a typo-laden excerpt from the Book of Job and then uses the theme of text manipulation and repair throughout the chapter. This is probably the best chapter in the book for two reasons: it gives you the material at a nice even pace and because the humor seems to work well here.

Chapter 3, 'The Gory Details,' gives just that. This is for the language lawyers in your shop, the ones who want to know all the ins and outs of a language (of which there are quite a few in *perl*) before they start programming in it. It includes sections on data types, operators, control structures, subroutines, regular expressions, formats (remember Cobol pictures?), special variables, and packages.

Chapter 4, 'Functions,' describes all the built-in functions in *perl* (over 125 of them). This includes quite a lot of the C library, which makes it somewhat lengthy (over 75 pages). Reading this section can improve your fluency in UNIX as well as in *perl*.

Chapter 5, 'Common Tasks in Perl,' is something of an odd chapter. It is like a cookbook full of recipes for things the authors think readers will want to do. It is a tutorial by example that demonstrates the power and style that *perl* can achieve. Some examples include: computing the difference or intersection of two arrays, squeezing out multiple blank lines, exception handling, and putting commas into numbers. These tasks vary quite bit; some seem a bit too specific to be useful. They are uniformly interesting, however, for their brevity.

Chapter 6, 'Real Perl Programs,' consists of program listings for an assortment of different kinds of full-blown programs. Subheadings include database manipulation, *grep* programs, programming aids, text manipulation tools, interprocess communication (*perl* includes access to sockets), and system administration programs. The programs are pretty readable (as *perl* programs go), but one of them extends for three pages of code with nary a comment.

The longest of the Chapter 6 programs is a rewrite of the *passwd* program in *perl*. The new version performs much more extensive checking than the one that is delivered with your system is likely to do.

Chapter 7, 'Other Oddments,' seems to be all the miscellany that didn't fit in one of the other major chapters. The command line flags are explained, commons goofs for novices are pointed out, the *perl* debugger and libraries are overviewed, tips for optimization are given, the security-analyzing features of *taintperl* are explained, and there's even a section on *perl* poetry.

Appendix A contains a BNF-like syntax grammar for *perl*. Appendix B outlines the *perl* library, which includes a number of useful functions.

The glossary at the end defines a lot of technical terms pertaining to *perl*, programming languages, and UNIX in general. The Three Principal Virtues of a programmer (according to Wall) are laid out: laziness, impatience, and hubris. Most of the entries are funny; some of them perhaps a little too much so, but it's still fun to read.

The 20-page index works well if you know the precise name of the topic you are seeking. However, if you're looking for help and hints, it is not quite as useful (e.g., if you want to convert from the UNIX system's seconds-since-1970 date to a human readable one, you must know *ctime* is the answer before you can find it).

Whether you can actually learn *perl* from this book is unclear. It depends upon your background: if you know UNIX shell programming already, *perl* will probably be easy. If you know C in a UNIX environment, it'll be even easier, but the differences between *perl* and C may annoy you. On the other hand, if you are neither a *sed* nor *awk* wizard and C programming is a black art to you, then this book probably isn't going to help you much. The more you know about UNIX to start with, the faster *perl* will make sense to you, and vice versa. Unfortunately, if you're still a UNIX novice, it'll be a lot harder for you, because the book assumes that you know about block structure (i.e., C's and *awk*'s braces), regular expressions, and the style of UNIX system calls.

Nonetheless, as the only book on *perl* available, and because it was co-written by the language's author, it is the definitive reference on the language and essential reading for anyone wanting to program in *perl*.

Tom Christiansen is a software development engineer at Convex Computer Corp. Tom also teaches a USENIX *tutorial on programming in perl. Rob Kolstad is software manager at Sun Microsystems in Colorado Springs, gives numerous tutorials on systems administration, and serves as Secretary of the* USENIX *Board of Directors.*

An Update on UNIX–Related Standards Activities

Jeffrey S. Haemer

Report Editor, USENIX Standards Watchdog Committee

ANSI X3B11.1: WORM File Systems

Andrew Hume <andrew@research.att.com> reports on the January 22–24, 1991 meeting in Murray Hill, NJ:

Introduction

X3B11.1 is working on a standard for file interchange on write-once media (both sequential and non-sequential, i.e., random access): a portable file system for WORMs. First let me apologize for laggardly snitching; we have had an extra meeting (in December) to accelerate our progress with the draft proposal, and I have been busy writing a programmer's guide to the draft proposal. I shall describe the results of the last three meetings, October (Nashua, NH), December (Murray Hill, NJ), and January (San Jose, CA), not in chronological order, but rather as a summary of where we are now. Although many details remain to be ironed out, we have broad agreement on the current proposal.

Multi-volume file systems

The draft proposal supports multi-volume file systems. To avoid the confusion that reigned at our meetings, I will define what this means. A *volume* is a logical address space (on some medium). Thus, a typical WORM disk is two volumes, as each side is addressed separately. A *volume partition* is simply a contiguous subset of a volume's address space. A *logical volume* is simply a set of (volume) partitions upon which a file system is recorded. Finally, a *logical volume set* is a set of volumes with a single volume set identifier. (That is, it is simply a publishing concept.) Note, however, that when I say file system, I mean a set of files and directories described by possibly multiple directory hierarchies (typically each would be in a different character set). The (logical) block size, not the physical sector size, is 2^i bytes, $9 \le i < 65536$, and implementations would have to support at least a block size of 64KB. The various size limits are generous; internal block addresses allow 64K volumes, 64K partitions per volume, and 2^{32} blocks per partition.

Volume Headers

The location of the volume header (the analog of the superblock) is a tricky issue because of the requirement that systems be able to boot off a disk in our format and there is simply no consensus on the size or location of the boot area. Accordingly, pointers to the volume header (actually a sequence of various descriptor records) are recorded at one or more of 0, 16, 64, 128, 192, 256, N - 16, N - 4 (where N is the size of the disk). The seek speed (or rather the lack of seek speed) of WORM disks encouraged us to put these at both ends of the disk. The volume header record, like all the other major control structures, has a 16-bit CRC and a unique 8-byte tag, which should prevent misrecognition.

Volume/Partition Structure

The volume layer handles space allocation for the volume, definitions of partitions, and bad-block mapping. The partition layer does its own space allocation, supports the file system, and does partition-access logging. Partitions have file-system-type tags; the intent is to allow partition w to be an X3B11.1 file system, partition x to be a CDROM file system, partition y to be an MS-DOS floppy file system, and partition z to be of unknown type. There should be a registry for this type field; vendors may want to register their file-system formats.

Bad-Block Handling

A simple defect-management scheme has been adopted; it is similar to the bad-block remapping scheme used for most SMD disks. There was considerable resistance to such a scheme, particularly from the representatives of the hardware vendors, as the (SCSI) WORM disks already do as much error detection/correction as is possible. However, defect management (above

the disk driver level) is still necessary because: 1. error correction/detection in the drive can be, and for performance reasons often is, turned off, 2. errors can easily occur between the disk and the host's main memory (have you ever heard of DMA or bus errors?), and 3. even though SCSI disks present an "error free" interface, most drives have a limited number of errors they can cope with, and many early drives did little or no error correction.

FCB Format

As you may recall, multiple versions of the *direct entry* (the equivalent of the inode) are stored in a data structure called the file control block (FCB). The original proposal involved various levels of indirect blocks exactly like classic UNIX file systems. We adopted my proposal (adapted from an observation by Dennis Ritchie) for a simpler, more general format that allows arbitrary structures, which can be specialized for different applications.

Partition Access Records

This is more like logging changes to the file system than a security thing like access control lists. The idea is to have periods of writing to the partition bracketed by specific control records so that it will be possible to tell if a system closed out that partition gracefully. (More bluntly, did we unmount the partition gracefully or did the system crash in the middle of a session?) These records are kept on a per-file-system basis and are recorded as variants of direct entries in a structure identical to FCBs. Another side issue is support for a so called "stable" record, which is analogous to the proposed stable sync feature of BSD UNIX. (The control structures such as inodes and indirect blocks are written to disk, but the user's data may not be, yet.) This peculiar state avoids the need to run *fsck* (or its equivalent) on the disk but you still have to get the user's data from somewhere. [Ed: does anyone really need this "stable" state?]

Recording Directories

For performance reasons, it is proposed that directories, or rather the records (FIDS) identifying the files (and subdirectories) in that directory, be kept in optionally sorted order. This would be in binary and not lexicographic order (thus evading nettlesome character-set-collating-order issues). It is not trivial to support this but is probably worth it. Related to this is the issue of system areas in directories and FIDs. It is expected that these areas will contain accelerator structures, such as B-tree indices and so on. Here and elsewhere in the standard, the governing principle is to allow systems to use such structures, but to neither mandate nor standardize their use.

Anonymous Files

There are numerous FCBs, or file-like objects, that have no FID. An example might be a Macintosh resource fork. The question is whether to make these visible to the user. This is a serious issue, and one not confined to this standard. It is an issue for the system supporting access to the file system on the disk. Do we rely on this system to do the right thing or should we mandate a mechanism? For example, take the example of a Macintosh file (with its resource fork) on a system (say UNIX) that doesn't have that concept. We can either trust that the vendor supplying your UNIX has implemented an *fcntl* (or *ioctl*) to access the resource fork, or we can evade the issue completely by mandating that the resource fork be available for normal access by a reserved name such as foo.RFORK. The general feeling is that users will not allow a standard to reserve parts of the file name space for its own use. Thus, it seems likely that access would have to be via standardized *fcntl* calls, but these are outside the scope of our standard.

Byte Order

I have pressed the issue of the byte order for numeric fields. The previous notion was to allow the recording system to choose the byte order. The issue is not technical (everyone seems happy to pick just one and stick with it) but political. We picked LSB order: the order used by the low-end (and slowest) systems. We measured the performance degradation for low-end MSB systems (the slowest Macintosh we could find), and the CPU cost of straightforward C code. Interpreting the byte order for the worst case (a block of integer block numbers) was about 10ms — comparable to doing a single disk I/O and one or two orders of magnitude less than the cost of doing a disk seek. (Careful assembly code would be much faster than this.)

Extended Attributes

The direct entry for a file has many attributes or fields. Some of these will be faster to access and be stored directly in the direct entry. The rest will be stored in an extended attribute record area much like resources in a Macintosh resource fork. There are two issues: which attributes get faster access and how do you access the other attributes? The former is something the standard specifies; our guiding principle was to include the fields needed for a UNIX *stat* or an MS-DOS (or VMS) *dir* command. Unfortunately, the issue of access is beyond the domain of our standard and needs to be addressed by POSIX, probably best by 1003.8. Internally within our standard, the extended attributes are identified by a 32-bit number, some of which are set in the standard and the rest by a registry maintained by some authority (like ANSI). The current list of extended attributes is given below. Treat it as very preliminary and subject to change.

information creation	file abstract
information modification	file type
information expiration	associated file
information effective	data compression
file creation	protection
file access	application-specific data segment
file modification	implementation segment
file backup	escape sequences segment
file expiration	action history
file attribute	icon
file effective	environment type

Character Sets

We have adopted a somewhat simpler way of dealing with character sets than the CD-ROM standard (ISO 9660). The current schemes available are

0	0–9A–Z_. from Latin-1 (ISO 8859–1),
1	portable filename character set 0–9A–Za–z_.- (POSIX 1003.1),
2	G_0 set from Latin-1,
3	all graphic characters from Latin-1, and
255	defined via escape sequences — the full scale mechanisms of ISO 2022, which are only rarely implemented.

International Activity

The appropriate ISO committee (SC15) has been reconstituted with Japan supplying secretariat duties. A meeting is expected in July or September and it is hoped that there will be close cooperation between X3B11.1 and SC15. There is some concern that ANSI might awaken the long-dormant file structure committee and that this might delay acceptance of X3B11.1's work. Also, because of a request by a working group involved in the Philips CD-WO device (a combination medium that is a 5.25in WORM with a CD-ROM portion), ECMA might also reconstitute its file structure committee (TC15).

Finale

What can, or should, you do? As always, I welcome any feedback, specific or general, on the work our committee does. (I must express my appreciation to USENIX for publishing these reports; nearly all the mail I have received about X3B11.1's work starts off like, "I read your report in the so-and-so issue of *;login:*.")In particular, I invite comments on any fields or attributes you would like standardized and — perhaps more important to the UNIX community — how to access auxiliary information about a file in *"a standard way."* Plenty of ad hoc solutions already exist for the cases of versioned files: VMS file systems on Ultrix systems, Macintosh files mounted as NFS file systems, and CD-ROM file systems. The number of these problems will certainly increase over time; we need to address the solutions now before we standardize on file system interfaces (such as 1003.8) that omit such mechanisms.

If you would like more details on X3B11.1's work, you should contact either me <andrew@research.att.com>, (908) 582–6262 or the committee chair, Ed Beshore <edb@hpgrla.hp.com>. I think the two most useful documents are the current draft of the working paper (about 80 pages) and a programmer's guide to the draft (about 12 pages written by me). I will send you copies of the latter document; requests for other documents or more general inquiries about X3B11.1's work would best be sent to Ed Beshore.

P1003.17 - Name Space/Directory Services (plus 1224/1224.1 Object Management)

[Editor's note: "Object" and "objection" have the same root word. What follows are three distinct viewpoints on TCOS's object-management activities. The first is Mark Hazzard's overview of 1003.17. The second is Scott Guthery's critique of the object management work, currently being jointly done by 1003.17 and 1224. The third is Enzo Signore's rebuttal of Scott's position. After you read them, you might want to let the committees know how you feel, either directly, or through Peter Collinson, the new USENIX Institutional Representative.]

Mark Hazzard <markh@rsvl.unisys.com> reports on the January 7–11, 1991 meeting in New Orleans, LA:

Introduction

New Orleans was busy for the P1003.17 — Name Space/Directory Services group. It was our first meeting as an "official" POSIX "dot" working group, and seemed to build on the momentum gained in the previous meeting. A good turnout from the old "core" group, coupled with infusion of "new blood" from the X/OPEN base-document development team, seemed to provide the right chemistry for some dynamic interchange and good solid progress.

As I stated last time, our group is currently in the process of "POSIXizing" XDS. This means reworking XDS to conform to POSIX style, content, and format requirements. Much of this is busywork that falls largely on the shoulders of our (overworked) Technical Editor. A first cut at the new format will be included with the first mailings. It can be best characterized as a "very preliminary pre-draft," and is intended to be a baseline from which a working draft can be built.

Language Independent Specification

A good deal of time was spent on LIS issues, both in our working sessions and in the joint working sessions with P1224 on common Object Management API issues. We were able to produce complete LISs for several functions and their data types, by building on the homework done by group members between meeting cycles. Readers may want to review the complicated discussion from last time on how and why two specifications, XOM (Object Management) and XDS (Directory Services), are required to form a single API to directory services. XOM is also used by the API to X.400.

Test Assertions

Several group members had fun finding out how to write test assertions for the C-language binding of our API. We even got together with some P1224 folks and worked on TAS for OM. We managed to write a few assertions and uncover some issues along the way. We also agreed to use identical conventions in .17 and P1224. During the process, we discovered that writing TAS is not a well-understood art, and what everyone seems to be doing is looking at what everyone else is doing.

Where do TAS go? They could be included with the function specification (possibly less work) or lumped together into a separate chapter or annex (possibly more work). We've opted for the lump. The rationale for this seemingly irrational decision is documentation page count ($$$). We figured that the only people who really care about test assertions (besides us standards types) are vendors, test suite writers, certification labs, and a few LARGE customers, like the U.S. Government. Everyone else (users) just wants to buy documentation on a certified API. We wanted to make it *really* easy for the IEEE to print "with" and "without" versions of the standard.

Object Management

"Object" and "management" are two intensely overloaded words. Used together, the two can instill fear in even the most seasoned hack. While conjuring up a name to put on the Project Authorization Request (PAR) for our common OM API, the combined talent of the .17 and 1224 groups decided that the best defense was a good offense and selected what may be the most offensive project title in the history of IEEE PARdom: "Standard for Common ASN.1 Object Management API for X.400 and Directory Services APIs." If approved, it should get a number like P1224.1 or something like that.

Flushed with success, the group decided to tackle the Scope section of the PAR, which prob-

ably constitutes its only real "meat." After considerable debate the group came up with this statement:

> The standard will define an ASN.1 Object Management (OM) Application Program Interface (API) for use with, but otherwise independent of, the X.400 and Directory Service (DS) APIs, which are currently being standardized. An application must be able to link and use multiple implementations of this API. This standard will provide language independent specification and "C" language bindings.

The words did not come without a little pain. The base document (XOM) was produced with specific targets in mind, namely the ASN.1-encoded objects and attributes defined in the XDS and X.400 specifications. It defines an API for manipulation of those objects across the API, but doesn't define the objects themselves. The object definitions are provided in the "primary" standard (either XDS or X.400) in a set of ASN.1 constructs called a "package."

In an accompanying article, Scott Guthery, a group member from the user community, expresses concern that there is no mechanism in the base document for extending existing objects or adding new ones. This is because the object definitions are well-defined within the context of their API (package) and have been hard-wired into the object manager.

Vendors can provide value added to extensions of their products, but users cannot. Further, a user who purchases a product from one vendor that uses a (non-standard) extended package will have no guarantee that it will work with an object manager from another vendor. With the ability to modify or create new packages in a standardized way, these problems could be avoided.

Counter arguments primarily addressed practical limitations to the scope, and the technical infeasibility of dynamically altering packages (which are really protocols). See Enzo Signore's accompanying article for a brief summary. The ability to extend an object package is not required for basic interoperability or portability for XDS or X.400 APIs as currently specified. A general-purpose user-extensible object management facility may be useful, but might be technically infeasible (or at least very difficult). It would almost certainly delay acceptance of APIs that depended on it.

Getting back to the PAR, the group agreed that the words in the scope addressed the immediate issue of getting an OM specification out so that P1003.17 and P1224 could continue. At the same time, the scope doesn't shut the door on a more general-purpose object manager, if it's deemed necessary and do-able.

I expect this will get sorted out after our next meeting in Chicago, but if this continues to be an area of high controversy, you'll see the topic resurface in my future reports.

In any case, the OM PAR was blessed by the Distributed Services Steering Committee and was forwarded to the TCOS SEC for further scrutiny.

Summary

So, that's a peek at what's going on in P1003.17. We can expect more of the same next time. We'll review our progress on LIS, probably do more test assertions, and generally begin to add some flesh to the document skeleton. We plan to meet with P1224 for a day to continue our co-development effort on common API to object management.

Scott Guthery <guthery@asc.slb.com> reports on the January 7–11, 1991 meeting in New Orleans, LA:

Here Come the Objects

X.400 (P1224) and Directory Services (P1003.17) have as their base documents X/Open documents, which in turn share an X/Open Object Management specification. At the just-concluded New Orleans POSIX meeting a Project Authorization Request (PAR) for a POSIX Object Management standard was formulated. Here is the scope of the PAR:

The standard will define an ASN.1 Object Management (OM) Application Program Interface (API) for use in conjunction with but otherwise independent of the X.400 and Directory Service (DS) APIs, which are currently being standardized. An application must be able to link and use multiple implementations of this API. This standard will provide language independent specification and "C" language bindings.

"What does that mean?" you may ask yourself. Based on discussions during the formation of this PAR the following is my understanding.

The first sentence says that object classes will be hard-wired into the OM and that the object managers being considered will only instantiate X.400 and DS classes. Further, only vendors of standard-conforming software will be able to add classes to the OM; there will be no provision on the standard interface for doing so. Finally, an OM will manage only instances of classes (objects) that are hard-wired into itself. Not surprisingly, this requires the second sentence.

The second sentence says that while the vendors are willing to agree on the interface, they are not prepared to agree on standards for objects themselves (even though they are all ASN.1-based). That is, vendor A's objects cannot be managed by vendor B's object manager and vice-versa. Objects themselves, as manipulated by the object manager, are to be proprietary. This is primarily because many of the vendors have already written object management software and the software that uses it, and are primarily interested in formulating a standard to which they can, after-the-fact, claim conformance.

The third sentence is boilerplate.

A couple of things bother me about this agenda. First, I don't like to see classes of users — privileged vendors who can define new classes vs. unwashed end-users who can only use what they're given (or, more properly, what they buy) — institutionalized in a standard.

Second, and really more bothersome (because I suspect the first one will work itself out naturally), is the "requirement" for multiple concurrently executing but not interoperating standard-conforming subsystems. My belief is that we should talk this one out carefully, make darn sure we all know exactly what we are talking about, ensure we are talking about the same thing, and convince ourselves it's something we want to enshrine in a standard.

Isn't this one purpose of a standard interoperation? If interoperation is left as an impedance-matching exercise for the user, is there really a useful standard in play even if the user can use a single interface on which to do the required impedance-matching? Might the jaundiced eye view this as a truck-sized hole through which vendors can drive claims to standard-compliance while exhibiting little-to-no effective standard-conformance behavior?

"Link and use multiple implementations" isn't good enough. Indeed, it's a bad idea. To me, it's analogous to a hardware standard (like RS232) specifying little more than implementations "use blue wires." I have to string a different set of blue wire for each vendor whose devices I purchase. And, what's worse, it's up to me to somehow get the information off one vendor's wires and onto another vendor's wires if I want the two vendors' devices to cooperate. The standard says something like "You get the information out at the end, which shall have 1/2 inch of bare wire." Frankly, being able to buy blue wire in bulk is little consolation for the trouble that I have to go to to make the whole mess work.

Of course, what I'm being invited to do is buy devices from only one vendor, which is, I suspect, exactly what the vendors had in mind when they put that "requirement" in the PAR. As an historical note, the second sentence originally started off "Users require that ..." until one of the few users around the table pointed out that single-source and vendor lock-in was not high on his list of requirements at all and expressed surprise that the standards process was or could be used to encourage it.

As they say in Norway, there's owls in the bushes.

Enzo Signore <enzo@retix.retix.com> reports on the January 7–11, 1991 meeting in New Orleans, LA:

Scott Guthery doesn't like the proposed 1003.17/1224 approach to Object Management. I do. Here's a summary of why I think Scott's objections miss the mark.

Since a package is another way of representing a protocol (a set of ASN.1 productions) the addition of another package to the API or the addition of new classes to the provided API implies defining extensions to the protocol. Aside from the feasibility of doing so, it would require the underlying service to be able to interpret the additional ASN.1 properly and to be able to encode and decode it. Unfortunately, it is not possible to do so in an implementation-independent way, since the OM representation of an object, even though it follows the ASN.1 skeleton, does not allow the service to generate a unique ASN.1 production. Said in different words, even if the client

application defines a new object class with some attributes (let's say of primitive types — booleans, integers, etc.) the sole object table does not allow the service to generate ASN.1, since all the context-specific tags and the notion of SEQ vs. SET are missing.

Therefore, designing such a new interface will:

1. prove wrong when the protocol cannot be extended;

2. be excessively complex to define because of OM design;

3. require overly sophisticated machinery in the service to be able to deal with generic and extensible object definitions.

1003.9: FORTRAN Bindings

Joseph J. King, Ph.D.<JKing@GCG.Com> reports on the January 7–11, 1991 meeting in New Orleans, LA:

POSIX is a set of portability standards that will span a diverse set of architectures such as VMS, UNIX, and OS/2. The FORTRAN binding to POSIX system services is nearing approval. In this report I'll discuss the current state, including the relationship of language-independent POSIX standards to the FORTRAN language binding, and the possibility that the POSIX/FORTRAN binding will be rejected by the International Standards Organization (ISO).

Portable Operating System Interface: POSIX

A POSIX standard is one of a group of standards being developed by the Institute of Electric and Electronic Engineers (IEEE), in cooperation with the International Standards Organization (ISO). The primary mission of these standards is to define a portable user and application environment. The POSIX development effort is currently subdivided into 19 separate numbered efforts — 1003.0 (POSIX Guide) through 1003.18 (PEP). Taken together, these groups are forming operating system standards in areas that range from networking to real-time. Half a dozen additional groups, also supervised by the IEEE's Technical Committee on Operating Systems, are creating related standards in areas like windowing toolkits. While POSIX started with UNIX as a model, POSIX standards are not limited to UNIX. For example, DIGITAL has announced a program that will incorporate some of the POSIX standards into VMS. Once adopted and implemented, POSIX standards will define a broad range of compatibility both within the UNIX family of operating systems and between other operating systems.

POSIX and FORTRAN

What follows is the January 1991 report on the progress of one of the POSIX working groups, POSIX.9. POSIX.9 is responsible for defining FORTRAN interfaces to the POSIX functionality defined by the other working groups. As a member of this committee I need to keep track of the progress of other committees to anticipate the next set of interfaces we will have to develop. At the moment there is only one published POSIX standard, which is referred to as POSIX.1.[1] POSIX.1 defines the functionality and C interface to POSIX operating system services. POSIX.9 is currently in public review with a standard that defines FORTRAN interfaces to the POSIX.1 system services. In addition to providing interfaces to system services such as process creation and interrupt handling, the draft also defines interfaces that will improve FORTRAN application portability and interoperability. For example, the draft contains procedures for reading the command line arguments, performing stream I/O, inheriting open files, getting the time of day, access to system constants, access to system structures, and performing bit operations.

"Thick" versus "thin"

The FORTRAN binding to POSIX is referred to as a "thin" binding. That means that it defines the FORTRAN interfaces to access the POSIX system services, but does not define the functionality of those services. Instead, the FORTRAN binding references the POSIX.1 standard for the functional definitions. The Ada binding to POSIX is also nearing completion. It is a "thick" binding, in that it defines both the Ada interfaces and functionality.

There are advantages and disadvantages to each approach. Thick bindings are easier to read, since all the information required is contained in one document. Also by using the thick approach

1. First published as IEEE 1003.1-1988, this standard has now been revised and updated, and achieved international status as ISO/IEC 9945-1 : 1990(E).

it is easier to map the functionality into native-language constructs. The Ada-bindings group has done just this and has been praised for producing a binding that is very Ada-like (as opposed to C-like).

Thin bindings constitute a more conservative approach. Since functionality is not defined in the thin binding, there is no opportunity for errors or inconsistencies to be introduced. Also, thin bindings are easier to adapt to changes in the base document. For example, the FORTRAN binding currently references the 1988 version of POSIX.1. Recently, however, POSIX.1 has been updated (1990) with several changes to functionality. After careful analysis at the January meeting, we determined that the FORTRAN binding requires only one substantive change to reference the 1990 standard as the base document.

ISO Requires Language Independence

The International Standards Organization (ISO) at one time required all POSIX functionality to be specified by language-independent standards. These are standards that specify functionality without specifying interfaces or syntax. Thin binding standards are then produced for each language to provide access to the functionality. In the last year ISO has relaxed this restriction to allow thick C bindings that define new functionality, but has excluded all other language bindings that do not reference a language-independent standard. Even though the FORTRAN binding is a thin binding, it is based on the thick C binding and not a language-independent specification as ISO requires. This is because there is no language-independent specification and such a specification could be a year or more away.

As a consequence, our working group will forward our draft for IEEE and ANSI processing when our work is complete. We will also ask ISO if they wish to adopt the IEEE standard at that time. This will give ISO another chance to say yes or no. We hope that they will adopt our binding at that time. If not, it may be several years before a language-independent standard is developed and we can produce a binding to it. We feel that our binding has usefulness in the FORTRAN community today, so that an ANSI standard, even in the absence of an ISO standard, would be useful.

Other issues

Other issues discussed at the January meeting included Fortran 90, the ballot process, and testing. There was some discussion of whether the POSIX.9 draft standard was Fortran-90-compatible. Since the FORTRAN binding to POSIX only requires FORTRAN 77 features it was agreed that our binding should be compatible with Fortran 90 compilers. We will look into this more carefully; however, after reviewing the areas in which Fortran 90 defines aspects for FORTRAN 77 that were previously undefined, I am confident that there are no conflicts that would prevent our binding from executing properly in a Fortran 90 environment.

I presented a short summary of Fortran 90 features to the working group. There was a discussion of which Fortran 90 features might be used to increase the usability and portability of the Fortran binding. There was interest in using derived types and user-defined operators to create an unsigned data type for Fortran — complete with the necessary mathematical operations. There was also an opinion that we should limit the Fortran 90 features we use to those already in existence in common practice (e.g., structures and Include). This would have the advantage that our Fortran 90 binding would not require a full Fortran 90 implementation and the disadvantage of not making the most of Fortran 90 features.

When this is printed we will be processing public ballot comments. The IEEE procedures for processing these comments was explained to us at this meeting. In order for our balloting to be successful, the following criteria must be met:

1. We must receive at least 75% of the ballots sent out, and

2. 75% of the yes-plus-no total must be yes.

Ballots received will be of one of three types, yes, no, and abstain. If there are any "no" votes, we are required to send out the objections to all those in the ballot group. They will then have the opportunity to change their vote. We will make changes to the draft and repeat this process until the necessary 75% is met and there are no new objections.

We discussed writing test assertions for our current draft. These assertions are used by an

implementor to prove conformance to the standard. It was agreed that since the FORTRAN bindings is a thin standard, our test assertions would be a thin document.

Work to be done

There is still much more to be done. At our next meeting we will be processing the public ballot. We hope to have a diverse range of opinions, and that an active balloting group will improve the quality of the standard. In this way, problems can be detected and fixed before they become part of the standard. If all goes well, that could be as soon as December 1991.

Our next meeting will be in Santa Clara, CA in July, and you are welcome to attend. Please contact either John, Loren, or me for details.

John McGrory (Chair)
Hewlett-Packard
19447 Pruneridge Ave
Cupertino, CA 95014
mcgrory%hpda@hplabs.hp.com
(408) 447-0265

E. Loren Buhle, Jr., Ph.D.
University of Pennsylvania School of Medicine
Rm 440A
3401 Walnut Street
Philadelphia, PA 19104
buhle@xrt.upenn.edu
(215) 622-3084

Joseph J. King, Ph.D.
Genetics Computer Group
575 Science Drive, Suite B
Madison, WI 52711
JKing@GCG.Com
(608) 231-5200

1003.5: Ada Bindings

Jayne Baker <cgb@d74sun.mitre.org> reports on the January 7-11, 1991 meeting in New Orleans, LA:

Introduction

The Ada Language Binding to the POSIX 1003.1 Base Specification (P1003.5) is moving through the IEEE ballot process.

Now that ballot resolution has begun, the P1003.5 working group no longer exists; we are now the P1003.5 Ballot Resolution Group. We spent this meeting doing just that — ballot resolution — addressing issues from our first ballot, chapter by chapter. This is no small task, so we also held an interim meeting in February.

This report outlines some issues from the first round of balloting, and touches on both potential future Ada work and our attempts to spread the P1003.5 word.

Ballot Resolution Issues

The following is a list of those who are responsible for specific sections of the P1003.5 binding:

Mitch Gart/ Alsys	Section 1	General
Steve Schwarm/ DEC	Section 2	Terminology and General Requirements
Ted Baker/ Florida State	Section 3	Process Primitives
Steve Deller/ Verdix	Section 4	Process Environment
Jim Lonjers/ Unisys	Section 5	Files and Directories
Mitch Gart/ Alsys	Section 6	Input and Output Primitives
Steve Schwarm/ DEC	Section 7	Device- and Class-Specific Functions
Jim Moore/IBM	Section 8	Language-Specific Services for Ada
Dave Emery/ MITRE	Chapter 9	System Databases

Below are some things I found particularly interesting in the discussions they led.

Naming Convention Issues

One global issue, more editorial than technical, but one that drew many ballot comments and objections, was our lack of a procedure-and-function-naming convention. We left names to the individual chapter authors, promising ourselves that we would address this issue later in document development. We never really did, but Bevin Brett/DEC proposed a naming convention before New Orleans via electronic mail, with these guidelines:

> Eliminate multiple names with same meaning by picking one (e.g., change "bad," "invalid," etc. to "bad");

Eliminate lengthy prefixes (e.g., change "POSIX_Process_Primitives_Process_Template" to "Template");
Make the parameter and subtype names the same.

Unfortunately, he also restructured packages liberally. We worried about making such sweeping changes after balloting had begun, but adopted several of his recommendations. Bevin agreed to revise his proposal, incorporating the changes accepted by the group. Chapter authors will evaluate the revised proposal. Accepted naming changes, supported by official ballot comments, will be incorporated into the document.

I/O Units

At least one balloter requests we use "byte" instead of "I/O unit," since an I/O unit is both "large enough to hold any member of the basic execution character set" and ">= 7 bits." Another goes farther, arguing that if "I/O unit" is exactly same as C's "char" we should eliminate confusion by saying so. A third balloter requests that POSIX, C, and Ada character sets be somehow unified.

Signals Discussion

Many balloters objected to signal semantics.

In the current, balloted draft (Draft 6) signals are interrupt entries and include semantics for interrupt delivery.

P1003.4a uses a SIGWAIT model with a single procedure equivalent to the Ada delay statement. Our current model may only support per-process signals, not per-thread signals, i.e., a subset of P1003.4a. Should we try to develop an approach to signals that does not break P1003.4a and P1003.4a/Ada?

We also need to review P1003.4's new signal proposal carefully to decide how much to change the P1003.5 binding to accommodate it. We do not know how balloters will react to such changes. Some balloters do not want signals tied to Ada tasks, because they want to use signals without using tasks. We could provide this, but others think signals without tasks is absurd. We will continue to work on this.

File Descriptor Types

File descriptor types are controversial. Balloters are divided on alternatives. Should file descriptors be private types? limited private types? integers? Each has its advocates. Here are sample arguments for making file descriptors private:

- Arithmetic operations are available for integer types. Does it make sense to perform additions on file descriptors?
- Sockets and P1003.4 memory-mapped files are used like file descriptors but may not be implemented as small integers, even in C;
- In V.4 the file descriptor limit has gone from 20 to some high number (4K?), making arrays of file descriptors less practical. (Array indexing with file descriptors is arguably bad C programming style anyway.)

Despite such arguments, we left file descriptors integer types because of the P1003.1 standard (note the definition of *dup(2)*). We noted in the rationale that we could have made file descriptors a private Ada type, required that they look like integers, and included special operations to disallow arithmetic functions on them, but the group thought that this would be too cluttered. File descriptors are acceptably close to integers.

Blocking Behavior

Tasks are a major source of complaints for UNIX Ada users. No one wants a whole process to block when one task within that process performs I/O or waits for a file lock. Unfortunately, P1003.1 doesn't support threads; blocking blocks an entire process, so our document provides support for per-process blocking. Should it also provide per-task blocking? Making implementors provide both behaviors could prove to be the wrong decision once POSIX supports threads.

After some discussion, two models for blocking were proposed:

- the knife-switch model, and
- the file model.

A knife-switch-model implementation elects either program blocking on I/O or task-level blocking on I/O for all files. Only one is supported. A file-model implementation selects either task- or program-level blocking for I/O on a per-file basis.

Mitch Gart suggested we relax our text in several places to allow alternative behaviors. He argued that separate predicates for blocking or non-blocking behavior are unnecessary because users will specify one or the other when files are opened; on the other hand, each system should provide an inquiry function that returns the blocking behaviors it supports.

We will add a statement to the normative conformance definition section (got all that?) saying that a strictly conforming application *shall not* depend on either task blocking or program blocking.

Future Work

Test Assertions for Current Work

To become an IEEE standard, the P1003.5 document must include test assertions. Jim Leathrum of Clemson University and his graduate students have volunteered to develop a set. We like this approach because the assertion writers will lack the working group's preconceptions and biases. Also, we won't have to do the work, and it frees the working group to press forward to other important POSIX/Ada tasks. We are well into the ballot process and will probably defer the test assertions to a separate document.

Ada Binding to Shells and Utilities (P1003.2)

Dave Emery continues to develop the Ada binding to P1003.2.

Ada Binding to Real-Time Extensions to POSIX (P1003.4)

Mars Gralia of Johns Hopkins University offered to put together an Ada/P1003.4 Project Authorization Request (PAR) broad enough to cover both the P1003.4 and .4a work. Ted Baker (former .5 snitch) generously provided the paper "Realtime Extension for Portable Operating Systems Ada Binding," some months back, which we will use as a starting point. We are currently refining a schedule to include in the PAR.

Spreading the P1003.5 Word in Europe

In my last report, I said that our group is making a real effort to educate people about the coming POSIX Ada Language Binding Standard. Dave Emery of MITRE submitted a team proposal for a tutorial at Ada Europe in Athens, Greece. It was accepted and scheduled as a half-day session for May 17, 1991. Unfortunately, we will probably cancel it because travel to Greece is currently discouraged.

POSIX Profiles

Jim Isaak <isaak@decvax.dec.com> reports on the January 7–11, 1991 meeting in New Orleans, LA:

The POSIX profile standards projects differ radically from the other POSIX activities. Most POSIX projects focus on specific interface specifications and, for the most part, on operating system services. The profile documents will neither define new interfaces nor be limited to operating system considerations.

What's a Profile?

The starting point on profiles is the grand experiment of OSI. By 1978, the OSI world had created a "complete" seven-layer model of communications and were ready to start developing standards that would populate that model. By 1988, over 140 standards had been accepted to fit into the seven layers. Any specific OSI implementation would pick some seven of these 140 standards, and specify any further options and parameters required by any of the standards.

The probability of two arbitrary, different OSI systems interoperating was nil. The solution advanced by OSI to this dilemma was to define a new kind of document — a *profile* — that specified the suite of standards, options, and parameters needed to meet a specific functional objective: indeed, profiles are also called "functional standards" (which says something about other standards).

The idea of profiles transcends OSI. For example, *de facto* profiles are commonplace. If you go into your local computer store and ask for "a PC with MS/DOS, Windows 3, a C compiler, and Lotus 1-2-3," that's a profile for your functional needs. Even the X/OPEN "Common Application Environment," which consists of several components described in their seven-volume X/OPEN Portability Guide, might be considered a profile.

although it is not clear that it would satisfy a specific functional objective.

The U.S. Government National Institute for Standards and Technology (NIST) introduced an "Applications Portability Profile" with their FIPS-151 in 1988[1], which has the same problem: the NIST "profile" is a collection of standards and other interface specifications with no focused functional objective.

POSIX and Profiles

P1003.10 is the first IEEE POSIX profile project, and its functional objective is more explicit: Supercomputing. The question this group is asking is, "what set of standard interfaces provides the complete environment supercomputing users need for applications portability, interoperability, and consistency of user interface?" Some standards identified so far are FORTRAN (F77), POSIX.1, GKS, and PHIGS.

The nasty word is "complete." It does not take much insight to see that even combinations of standards won't be complete enough for some sophisticated applications. Application environment profile groups also must identify areas where additional standards are needed. This is a second value of profiles: supplementing the "roadmap though the maze" goal of OSI profiles. At a minimum, profile groups should document requirements not addressed by the standards, to help vendors and users ensure missing capabilities are provided, even if they're not standardized.

For supercomputing, for example, performance requirements fall into this category. Sometimes, profile groups can do more than just document, though. What happens, for example, if the profile work reveals the need for an interface that isn't yet standardized? There are two possibilities. First, the profile group can tell an existing standards group working in a related area they need the new interfaces (and offer expert aid, if necessary). If that doesn't work, the profile group can try to spin off a new group or at least a new project. For example, the supercomputing profile work has already spun off two projects: the P1003.9 working group, which is providing FORTRAN interfaces to POSIX.1, and the P1003.15 project for batch services.[2]

Other POSIX profile projects already underway are transaction processing (.11), real time (.13), and multiprocessing (.14).

PEP: a prototype standard

Right now, profiles are being generated bottom-up, starting from real-world problems, leveraging existing standards, and exposing gaps. There is no formal model for applications portability, and few users are willing to wait for one. In an ideal world, with some well-understood formal model of a complete computing environment, standards might be generated top-down, much like the OSI approach. How do we grope our way to such a model?

At the international level, Technical Study Group 1 (TSG1), is just finishing recommendations to ISO on "interfaces for applications portability," which will include some modeling concepts for a top-down approach. The most solid recommendations coming out of this work will advocate the extension of ISO profiling work beyond OSI to address applications portability and to help identify requirements for standards work.

Industrial groups, like the Petrotechnical Open Software Company (POSC), are also interested in profiles (for "upstream" petroleum engineering applications). The European Workshop on Open Systems (EWOS) is also expanding their charter from OSI profiles to application portability. Much early groundwork for OSI profiles was developed by folks now in EWOS, and much of EWOS's current work is to establish a framework and a set of procedures for this larger problem space.

The IEEE's TCOS is taking a different approach: sort of rapid prototyping for standards. The most recently approved IEEE POSIX project is P1003.18, the POSIX Platform Environment Pro-

1. They have recently published an expanded view of this for public comment.

2. The "batch services" work isn't just batch processing. In the world of supercomputing, jobs can run beyond either the system's MTBF — mean time between failures — or its MTBSHPJTYOFAW — mean time before some higher priority job throws you out for a week. To handle this with some grace, applications "checkpoint" their state and can restart from that state. Extended services for checkpoint and restart are part of the .15 task.

file (PEP). A simplistic statement of its goal is, "to define the standards which together describe what folks have traditionally called UNIX": — POSIX.1 plus POSIX.2 plus the C standard.[3]

Its real goal, though, is to address three closely related needs.

1. PEP will provide a common foundation (platform!) for the more focused application environment profile projects (.10, .11, ...).

2. It will help all users of the POSIX standards to understand the line between the traditional "UNIX" space and any new work of the POSIX groups. This does not mean that PEP will not include some new work over time, but it will identify a useful stable ground (platform!) in the rapid evolution of POSIX standards.

3. Profiles are new to IEEE, ANSI, and ISO, and a simple example will help folks to get a handle on what profiles are all about. PEP provides a simple case, building on POSIX.1 (system interfaces), POSIX.2 (and .2a) (commands), the C language, and potentially Ada and POSIX.5, and/or FORTRAN and POSIX.9.

In effect, PEP will blaze the trail for other more complex profile tasks, like supercomputing or transaction processing, and will be a stepping stone (platform!) for those efforts, providing them a clearer path into ISO.

Profiles as Configuration Management Tools

The second point above may need explaining. Profiles take a snapshot of the standards at a point in time. Supercomputing is specifically selecting FORTRAN 77 because it matches today's needs. Fortran[4] will evolve. A future version of POSIX.10 may include a future version of Fortran. The array of standards is moving forward asynchronously, and often chaotically; profiles group together standards to provide a form of "release control" or "configuration management." An application and a system can refer to a specific version of a specific profile.

3. Of course, many readers will argue that this is hopelessly incomplete ("How can you have a UNIX system without *emacs*?"), but bear with me.

4. The next generation Fortran is properly presented in mixed case, by committee decree, whereas historically FORTRAN has been an acronym and all upper case.

Benefits Profiles Will Bring

Profiles provide an easy way for users, application developers, and vendors to describe environments. Application developers and ISV's will finally have a clear target-space for development. Profiles will tell customers what facilities the target systems for their software supply. Eventually, we will see conformance testing of integrated profiles, providing a higher level of confidence in the environment.

1003.6: Security Extensions

Ana María de Alvaré <anamaria@sgi.com> reports on the January 7–11, 1991 meeting in New Orleans, LA:

Overview

The P1003.6 group met for the entire week. Our main task was preparing draft 8 for mock ballot. We also planned for P1003.6 test assertions and discussed file locking, manipulating or duplicating the information in opaque data objects, and allowing *ps* to show privileges and MAC labels of processes.

We also heard two proposals at the meeting, one on Privileges and one on Discretionary Access Control, which I discuss in the relevant subgroup sections below.

Mock Ballot

P1003.6 plans to go to mock ballot after our April meeting. We will review comments at the July meeting, and try to ballot the document soon afterwards. The October meeting will be used for ballot resolution and clean-up.

To prepare for mock ballot, the working group submitted written comments on the current draft, and subgroups spent the week addressing them. Commenters included Chris Hughes (ICL), Roland Clouse (Unisys), Dan Ujihara (SUN), and me (SGI).

Test Assertion Plans

The group decided to create a separate test-assertions document that parallels the current document. Each subgroup will be responsible for its own test assertions, and will ensure that the assertions document and the main document re-

main consistent (i.e., any updates to the P1003.6 document will trigger changes to the assertions document). Dave Rogers of Data Logic and I are co-chairing this effort. If you are interested in helping to write test assertions, please let us know.

Opaque Security Data Object Duplication

Duplicating the information in opaque security data objects — ACLs, labels, and privileges — presents three distinct kinds of problems:

1. duplicating the information within a process,

2. passing the information between processes in a single system, and

3. exporting the information out of a system.

Copying the information within a process is simple. What's hard is copying it out of the process's context — for example, for backups. We decided that such exporting will require passing out both object addresses and sizes, as well as data characteristics, such as *binary*, *text*, or *function*.

Privileges

John Griffith (HP/APOLLO) presented a new privileges proposal that simplified determining whether a process has, lacks, or inherits a privilege.

In draft 8, a process could only inherit a privilege if the "allowed" file-privilege attribute was set: inheritance, through the inheritable group, depended on restrictions provided by the "allowed" file privilege attribute.

The subgroup agreed that this needed simplifying. The newly agreed-on substitute is that a privilege can be inheritable if it exists in the inheritable group or if the file's "forced" privilege attribute is on. In other words, after an exec occurs, a privilege that is on in the inheritable privilege group can turn itself on in the permitted privilege group.

The subgroup spent much of the remaining time editing its part of the document. Two issues I hope will be resolved next meeting are:

1. accommodating privileged shell scripts in the current proposal, and

2. determining how to store privilege information for later use.

Discretionary Access Control

The new DAC proposal consisted of two documents representing a collaborative effort by Paul Karger (OSF), Rand Hoven (HP/APOLLO), and Jon Spencer (Data General). It tried to simplify the way default ACLs and MASK_OBJs work, and it removed any requirement for MASK_OBJ entries when no additional ACL entries existed. In the end, we decided to retain the old scheme but will try to shore up areas that the new proposal pointed out were particularly weak. The proposal's sponsors agreed to this, providing the new draft offers a satisfactory alternative simplification.

The subgroup also attacked the opaque object issue described earlier, defining an interface to interconvert DAC opaque objects and text strings, and a relocatable ACL format that can be stored in an audit record.

The DAC subgroup will pass their draft to the full group after the next meeting.

Mandatory Access Control

The MAC subgroup discussed the written comments to their section and feel they will be ready for ballot after the next meeting.

Two major issues arose:

1. whether our document should address special (block and character device) files, and

2. whether we needed a *dup()*-like function to copy internal formats.

The subgroup decided the current version of P1003.6 shouldn't address terminals or other special files, but the second issue will be passed on to the entire group.

Audit

The Audit subgroup discussed all the written comments and will only need one more meeting to be ready for ballot. Their work, including mandatory record types, will be based on X/Open's. They will not address Portable Data Record Format, and optional record types will be implementation-defined.

Clearly, audit functions will need both pointers to objects and their sizes to operate on MAC, DAC, and Privilege opaque data. Because of this, I predict all three subgroups will have to provide interfaces to provide the information.

Liaison .6/.7/.8

The liaison group met again to discuss areas of compatibility and overlap between our respective documents. (The October P1003.6 snitch report sketches our ongoing agenda.) We identified areas that P1003.6 (Security), P1003.7 (System Administration), and P1003.8 (TFA) already handle, areas we might handle, and areas that are falling through the cracks. After we finish identifying areas of concern, we may write PARs for anything we cannot farm out to existing groups. In April, we will discuss how to report our findings back to the three groups.

X11 Release 5 Now Available

Gary Henderson
IXI Limited
Cambridge
United Kingdom

Gary is an experienced software consultant with over 4 years knowledge of working with the X Window System. Gary has hands-on experience of porting X and Motif to a variety of platforms and is also a seasoned toolkit programmer. His role at IXI involves lecturing in X and Motif programming as well as heading up the Motif Development Team with responsibility for IXI's Motif Development Kit for Sun product.

On 9 August, the MIT X Consortium issued to its members a new release of the X Window System, X11 Release 5. As these releases are only made every 18 months or so and contain significant new technologies, this is an important event for anyone working with X.

X11 Release 5 includes scalable fonts technology, support of internationalisation and the PHIGS international 3-D graphics standard. The latter is particularly important as the inclusion of the PHIGS Extension to X or PEX provides a standardised way of supporting 3-dimensional graphics. It has been possible to generate 3-D graphics on X as vendors have added their own extensions to the X server but this has been in a stand-alone form. This release marks the first time that there is networked support for 3-D graphics.

PHIGS or Programmer's Hierarchical Interactive Graphics Systems has been criticised in the past for its complexity and the demands on processing power to draw the 3-D graphics. However, as an established international standard, it was natural for the X Consortium to select PHIGS for the basis of the 3-D version. There are proprietary graphics libraries such as Silicon Graphics's GL which has achieve wide recognition to the extent that Network Computing Devices (NCD), the leading X terminal supplier, may be supporting GL alongside PEX.

The new PEX files take 10.5Mb memory and take the form of a server extension and client side library. The client uses two processes, one to handle X events and the other the user application.

The new font server technology widens the font repertoire for X users. A single font server sitting on a network node can provide a large selection of fonts in a variety of styles in all point sizes. Each point size of the different fonts are not stored in a separate file as in earlier X releases but as a single copy which can be scaled up or down as required. This considerably reduces font storage space and increases the range of point sizes available.

There are two sample font scaling implementations; one using bitmap fonts which allows existing X fonts to be scaled and the other using font outlines contributed by Bitstream Inc of Cambridge, Mass. The latter is said to produce better quality images but needs special outline font files. Bitstream supply a Charter outline font in normal, bold, italic and bold-italic.

Colour has now been standardised so that you will no longer get any colour variation when you move from one terminal to another. Originally, colour was represented as RGB values but in X11 Release 5, this system has been improved by a device-independent colour system at the Xlib level. Developed by a European standards body, the Commission Internationale De L'Eclairage, the CIE technology provides a standardised representation of colours from one display to another. Several methods of describing colours are implemented including CIE u'v'Y, CIE XYZ and Tektronix's TeKHVC systems.

A good deal of work has been invested into providing adequate support for Internationalisation. This is obviously key to the success of X worldwide and will no doubt lead to the Open Software Foundation and Unix International providing further internationalisation support for their GUI products - Motif and Open Look. In fact, the next release of Motif, v1.2, will be based on X11 Release 5.

The work takes into account the ANSI and ISO standards for internationalisation. The previous releases of X11 could only effectively deal with 8-bit character sets such as error messages

and the resource database and had no way of inputing the larger character sets. X11 Release 5 now supports characters used in Japan, China and other parts of the Far East.

Apart from these major new technologies, Xlib has been optimised - routines that the toolkit calls frequently have been improved some by a factor of 300. The server now supports true Save Unders, rather than saving the contents of the entire window, only the obscured section is now remembered. There is now support for hardware cursors for systems that have this feature.

The initial release of X11 Release 5 only includes the core technology (about 74Mb in size). However, the full release on 10 October will also contain user contributed software but be warned it will be rather large!

If you would like to receive your copy of X11 Release 5, then tapes are available from the X Consortium in Cambridge, Mass (price had not been decided as we went to press). Alternatively, IXI are supplying the MIT tape here in Europe at 295 pounds, for further information call IXI on +44 223 462131.

NOTE

X11 Release 5 is available from numerous sites across Australia, by anonymous *ftp* and ACSNet fetchfile. For details consult the newsgroup *aus.archives*.

USLE Column

Dr. Alan Brown
Principal Consultant, specialising in TUXEDO
USLE
London
United Kingdom

There has been much discussion about USL's Transaction Processing System, TUXEDO. Alan Brown joins us in this edition to talk about the product.

Alan Brown is Principal consultant at UNIX System Laboratories Europe based in London. He has responsibility for TUXEDO in Europe - this includes the provision of pre- and post-sales technical support, training and consultancy.

For further information on this column, please contact Gill Mogg on gill@eul.uucp. Gill is Marketing Manager at USLE.

The TUXEDO Transaction Processing System Release 4.1

Introduction

The trend today is towards using networks of high powered mini/workstation style machines as part of the Open Systems movement. To explain today's definition of distributed, open online transaction processing (OLTP) computing, I will begin with a short look at the major evolutionary steps that preceded it. (See Figure 1)

Starting with a batch transaction processing environment, demands for data integrity and availability became more and more pressing, leading to the need for online transaction processing. Businesses in which data that once was sufficient on a weekly basis now need it available every day; international companies need consolidated reports for world-wide regions to be drawn up from reports for separate offices; many businesses need to know instantaneously about their transactions. A transaction processing system can provide the framework for meeting these demands as it provides realtime data access and updates while combining multiple business systems into one coherent application.

The last few years have seen an ever intensifying move towards Open OLTP The benefits of Open Systems in general apply as well to a transaction processing environment. Open OLTP has the benefit of being portable and interoperable. Constrained MIS budgets leave many MIS executives seeking to level current investments with increasing demands for IT solutions and processing power. Looking toward OLTP, they might realise that their preferred solution consists of a mixture of hardware platforms, databases, networks, LANs, presentation managers and the like. Using open OLTP not only allows the interconnection of current products but also protects the investment through compatibility with future developments

This article will discuss the different components of the TUXEDO Transaction Processing System Release 4.1 developed by UNIX System Laboratories. The TUXEDO System is a mature, available OLTP product supported on more than twenty hardware platforms, by nine operating systems and currently deployed in over fifty applications. As the roots of TUXEDO lie in the UNIX operating system, it is an open OLTP system, supporting unlimited front-end interfaces, network protocols and resource managers. TUXEDO has been evolving since 1978 and is fast becoming recognised a the standard for open OLTP solutions. In particular, it offers the following key capabilities:

- TUXEDO provides application designers and programmers with a state-of- the-art framework for building OLTP applications.
- Openness at all levels of the TUXEDO System architecture with true heterogeneity across all layers of an OLTP application in combination with product components that adhere to standards giving the customer upward compatibility with future hardware and software investments, to take advantage of price/performance and yet still protecting past investment.
- Control over distributed data and functionality gives the user a unified view of a distributed application.

TUXEDO is an internationalised product that allows the user to be presented with diagnostic and system messages in their language of choice, reflecting national date, time and currency

conventions. This is as specified by the X/Open Portability Guide Issue 3.

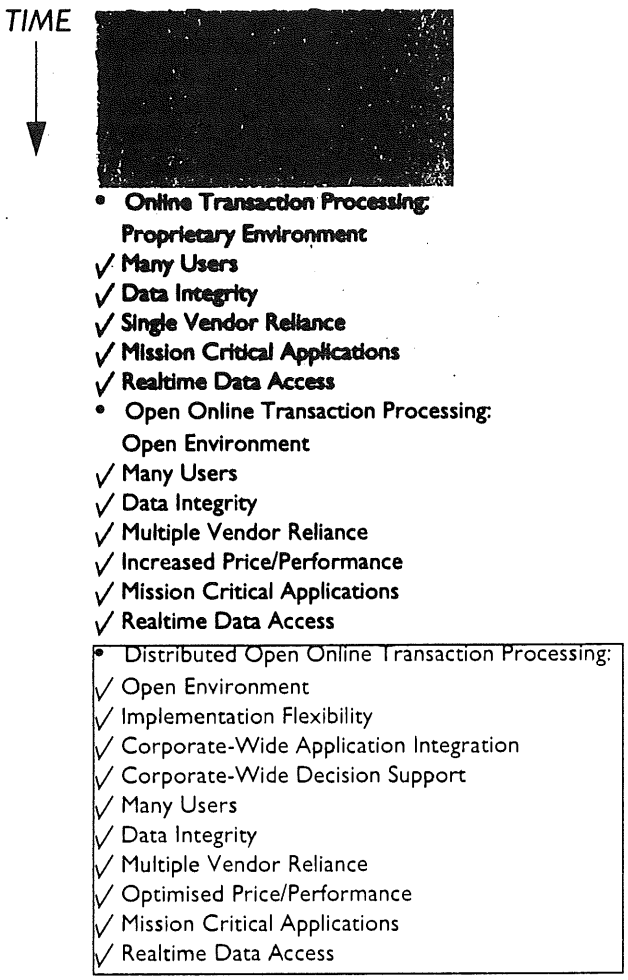


Figure 1: The move towards Open Online Transaction Processing

TUXEDO also gives an application writer a common software platform across a multitude of systems both UNIX-based and proprietary systems (e.g. VMS(TM), AIX., HP-UX, Sun Os(TM) and ULTRIX(TM)). This is achieved by using highly portable code based on industry standards including SVID, POSIX and XPG.

A modular approach allows the product components of TUXEDO to be networked and integrated into other OLTP products. The System comprises two basic parts, which can be licensed and deployed separately: the transaction manager TUXEDO System/T, and a high performance database management system, TUXEDO System/D. Two additional components will shortly be available: TUXEDO /WS, workstation extensions to System/T, and TUXEDO/HOST, enabling System/T to use mainframe services.

The following discussion will now look at the four TUXEDO components.

TUXEDO System/T

The Client/Server Model

The client/server model is the basic structure of the TUXEDO System, providing location transparency by mapping the logical name of a service to the physical address of the server that can perform that service. (See Figure 2)

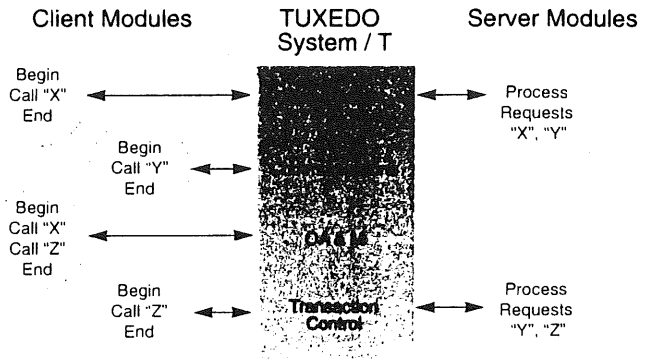


Fig : 2 The Client/Server model

Figure 2: The Client/Server Model

A client process collects input, makes service requests, receives replies to requests and puts out results. A server process accepts a service request, performs the work (which may include additional service requests for the original request), and, when it has finished, either returns results to the requester or forwards the results obtained so far to a new service. Server processes are stateless: several servers may offer the same service and repeated requests for the same service may go to different servers. The request/response model supports asynchronous and synchronous communications.

System/T provides a number of features that enhance the client/server model. These are described below. An application client will use a well defined interface called the ATMI (Application to Transaction Management Interface) to pass requests to a server process. Similarly the server will respond using the same ATMI. This ATMI heightens the underlying client/server configuration and transparently controls transactions on behalf of the client.

The Name Server: Bulletin Board

The Bulletin Board (BB) is the heart of System/T. One of its main functions is to map the service name a client uses to an internally maintained physical address to which a request can be sent. In that way, the BB allows clients to request services by name rather than sending requests to a specific address. Clients do not have to know which server handles their request.

The BB also keeps statistics to aid in load balancing when deciding where a client's request should be sent. This enables System/T to keep track of how many outstanding requests exist and to which servers they are destined. It then routes the request to the server most likely to process it first. The BB is implemented as a piece of shared memory to which all client and server processes attach. Consistency is guaranteed by a combination of user- and system-level locking.

Operations, Administration and Maintenance (OA&M)

System /T allows an application designer to centrally define the hardware, software and networking resources that make up an OLTP application. It can be stated where servers and services are supposed to run and where they should be migrated to in the event of a processor failure. In addition, various characteristics can be assigned to the application's software resources, including processor placement and scheduling information, process recovery criteria and time-out periods. System /T also provides

for central configuration management and dynamic tools for starting, stopping or administrate a distributed OLTP application.

Servers can be dynamically started or stopped making only selected services available. Various parameters such as time-out interval, priorities and load factors may also be changed dynamically. If a processor fails or needs maintenance, the server and services on the out-of-service processor can be migrated to another processor without interruption to the running application.

To enhance application availability, robustness features are built into System /T, including process viability checks, time-out checks, automatic server re-start and recovery procedures. In distributed and multiprocessing environments, System/T can increase the availability of an application by replicating servers and services across several processors. Application data can be partitioned across processors participating in the application and accessed by data-dependent routing of service requests. This enhances the distribution of services and the application's resiliency to select processor failures.

To ensure maximum throughput, System /T automatically performs load balancing and scheduling throughout the system. It uses per-service load factors and keeps totals on outstanding work to deliver a particular request to the server that can process it most quickly.

System/T applications can naturally extend over a set of machines on a LAN without special attention in the application. Communications are handled by System/T using the network independent library (TLI) or the sockets interface.

Transaction Control

Ideally, an open distributed transaction processing system imposes no restrictions on the application?s choice of RMs. Employing the XA interface defined by X/Open, TUXEDO System/T communicates with all resource managers that are XA compliant.

The distributed transaction comprises two general elements: the transaction manager handles the global part, it keeps track of local transactions participating in the global transaction and handles all commit and recovery decisions. The resource manager/server process controls the individual pieces, associates them with the identifier for the global part and carries out the decisions of the TM as they affect the local transaction. To participate in a two-phase commit, the resource manager (RM) must be able to start, precommit, commit and abort a local transaction.

The RMs that participate in a distributed transaction provide support for a two-phase commit presumed-abort protocol:

- They have to offer subroutines that begin, precommit, commit and abort a local transaction.
- They must not make independent commit or failure handling decisions as part of recovery for precommitted local transactions associated with a GTRID without informing the TM.

Each global transaction must have a Global TRansaction IDentifier (GTRID) which is unique across the OLTP system. The use of GTRID is required for recovery as well as a unique identification of distributed units of work. In a two phase commit protocol, a commit coordinator requests participants to precommit. When all participants have reported successful precommit, the GTRID, the coordinator id and a list of ids for RM participants must be logged. The GTRID includes the coordinator id and the RM must put it into stable storage. This allows the transaction manager to obtain a list of GTRIDS in case of a failure and request the status of the global transaction for the GTRID.

The function of the commit coordinator is the management of a two-phase commit protocol. TUXEDO System/T maintains internal information about the participants in a server process, and when an application requests the transaction to be committed, the coordinator of the transaction uses the internal information to begin the commit process by sending precommit requests to participants. The TUXEDO System/T code for commit services is written using common transaction manager/resource manager interface subroutines; different implementations of these subroutines, included in the respective RM libraries, are used to build commit servers.

TUXEDO System/D

TUXEDO System/D is a high performance database management system specifically designed to support the needs of an OLTP application running under UNIX System V. (See Figure 3, TUXEDO System/D architecture)It provides a set of tools to build and administer such applications by implementing an independent user level file system which stores database and transaction log information on a privately managed set of raw disk partitions. Since all the I/O is synchronous and bypasses the UNIX buffer pool, data from committed transactions is written to stable storage providing database consistency and recoverability.

Fig : 3 Tuxedo System / D Architecture

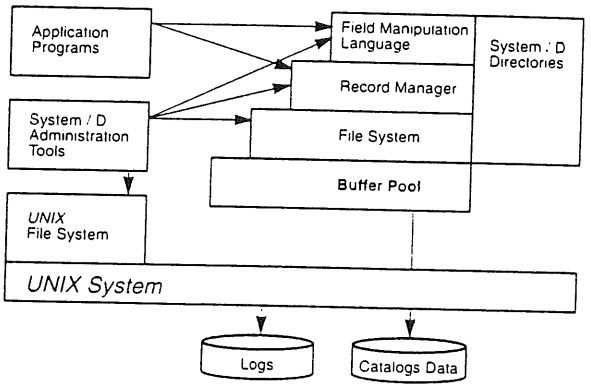


Figure 3: Tuxedo System/D Architecture

TUXEDO System/D is compliant to X/Open's XA interface definition. System/D uses a redo/no-undo strategy with updates logged to separate devices before being applied to the database. TUXEDO System/D also supports the use of embedded ANSI SQL. In case of a system failure, the system only has to redo the updates of committed transactions that are not in the database. High performance database access in System/D has been achieved through the extensive use of caching (using shared memory) optimised for DBMS access, extent-based disk allocation, support of multiple transaction consistency levels and by allowing programmers to navigate through the database with the use of a record at a time interface.

Enterprise Transaction Processing (ETP)

The product direction for the TUXEDO Transaction Manager beyond Release 4.1[1] will expand System/T to embrace proprietary, non-UNIX System V systems to provide for a truly heterogeneous environment. Enhancements will extend System/T client support to include the interaction of workstations and PCs

1. TUXEDO /WS and TUXEDO /HOST will be generally available from USL fourth quarter 1991

with System/T servers, and the addition of gateways to embrace servers within external OLTP systems. (See Figure 4)

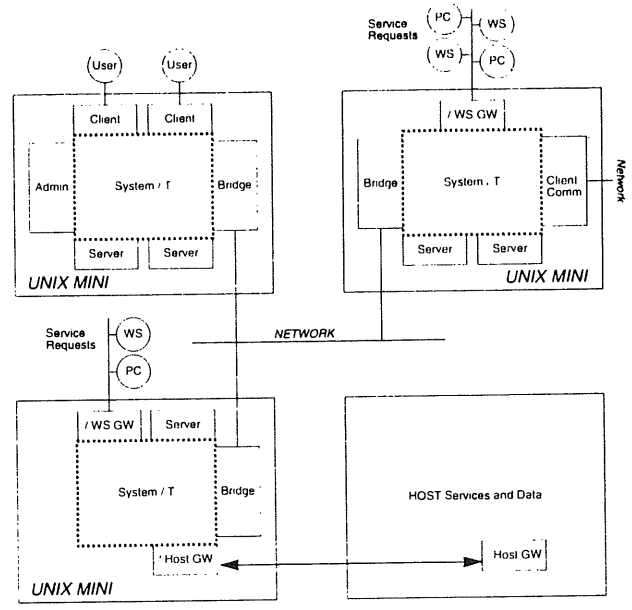


Fig : 4 TUXEDO Release 4.1 Enterpnse Transaction model

Figure 4: TUXEDO Release 4.1 Enterprise Transaction Model

Workstation Support - /WS

The workstation interface (/WS) will allow an application to use its existing base of workstations and PCs to develop and run System/T client programs. These client programs have access to System/T application servers throughout the network. This facility will provide the advantage of removing terminal character processing overhead from server processors and places the overhead on intelligent processors. It also allows application developers to use the presentation facilities and screen managers available for workstations and PCs. In addition to a full client programming interface, further security measures will be implemented to control access to application servers. Gateway software will provide all necessary functionality to establish the LAN connection between the client and server, manage communications and perform the necessary encoding and decoding of messages.

/WS can be utilised on both disk-based and diskless desktop computers that run either the UNIX System or MS-DOS. Full access from diskless systems will be dependent on properties of the network such as remote file access. The MS-DOS version of /WS will support the NETBios interface to the available network provider. The UNIX System version of /WS will support both TLI (Transport Layer Interface) and BSD (Berkeley Software Distribution) Sockets networking interfaces.

/WS will provide maintenance of investment in current systems and will allow the user to take better advantage of desktop power and hardware trends.

Host Support

Major corporate computing resources such as services and data reside on machines that currently run proprietary OLTP software. TUXEDO System/T's new /HOST feature extends the client/server model into the surrounding host environment and allows a System/T application to provide transparent access to remote, external services and data. System/T resident gateway servers represent host resident servers and services on the local System/T node and manage communications with the host computer. Through the gateway servers, host resident servers and services are viewed as being part of a single System/T application. Consequently, the availability of host resident services can be accessed through the standard System/T ATMI programming interface.

With /HOST, USL will provide a System/T gateway for the IBM MVS/CICS environment. The LU 6.2 protocol will be used to perform peer-to-peer communications between the UNIX-resident and the CICS-resident gateways. Software will be provided that will allow a UNIX System/T node to be linked via an LU 6.2 communications library to a CICS application programming interface and CICS administrative utilities. Data conversion will be performed by System/T for the ATMI supported data structures; user defined encode/decode functions will also be supported.

Summary

TUXEDO is a mature, open, on-line transaction processing system, developed to meet widespread market needs. Where applications writers are concerned, client applications can be written not only in C or Cobol but also using CASE tools and some 4GLs (e.g. Ally), resulting in increasing ease of application development.

The TUXEDO system is continually evolving as is demonstrated by the forthcoming release of /WS and /HOST, and with the increasing demand for OLTP in all sectors, TUXEDO will underly most of the TP systems in the Open Systems arena.

The system gives applications developers a great deal of freedom as it allows client applications to be written not only in C or Cobol but also in CASE and some 4GLs (e.g. Ally).

Book Reviews

Reviewed by Simon Kenyon of ICL - Information Technology Centre, Dublin, IRELAND <simon@itc.icl.ie>

X Toolkit Intrinsics Programming Manual OSF/Motif Edition

Adrian Nye and Tim O'Reilly
O'Reilly and Associates Inc., December 1990
ISBN 0-937175-62-5.
(US) Price $30, Soft Back, 632 pp.

This is volume 4 in the X Window System series published by O'Reilly and Associates.

The blurb for this book says that it is a "complete programmer's guide to the X Toolkit (Xt)" and that the book "uses the Motif widgets to demonstrate how to use existing widgets, but is equally applicable to and provides a good introduction to programming with any other widget set based on Xt, such as the MIT Athena widgets, or AT&T's OPEN LOOK widget set".

This aside, what is this book? It is a rehash of an earlier edition which described the Intrinsics using the Athena Widgets as examples. It starts off with a chapter describing what the X Window System is all about and then swiftly moves on to discuss the X Toolkit (Xt), which is described as "providing a simplified approach to graphical user-interface programming".

The book is structured as a tutorial. An example application is developed, becominging increasingly more complex as new features are added to it. The culmination of this is the recoding of the example using a new widget, the BitmapEdit widget.

The intention of the book is that the reader should become a proficient Xt programmer, and for my money this objective is achieved. The book arrived just as we in the ITC were about to port a large application from XView to Xt. We already had the previous edition of this book; but it was my copy that kept disappearing off my desk. This is perhaps more relevant than my subjective opinion.

I do have a number of gripes about this book however. The first is that the typography is awful. It really gets in the way of the learning process. For a book about constructing graphical user-interfaces, faking the screen images makes the images worse than useless. They give the reader no clue as to what the screen should actually look like.

The second point is that the running example leaves a few "exercises for the reader". It is like a lecture in mathematics, where some intermediate steps are left to the student. This caused me some grief.

The description of OPEN LOOK and Athena are sketchy and in the case of OPEN LOOK, out of date.

The final point is that why are all the interesting bits "beyond the scope of this book".

There are no glaringly obvious errors that I could see, apart from some problems with the resource entries accompanying the examples.

This is a book for the programmer who wishes to program with widgets, and Motif in particular. It is for someone who needs to know, as there is effort required. This effort is a lot less than wading through the MIT supplied documentation.

Recommended.

X Window System User's Guide OSF/Motif Edition

Valerie Quercia and Tim O'Reilly
O'Reilly and AssociatesInc.
December 1990,
ISBN 0-937175-61-7.
(US) Price $30, Soft Back, 709 pp

This is volume 3 in the X Window System series published by O'Reilly and Associates.

This book is an introduction to the X Window System. It describes the basic concepts of X and gives details of all the MIT supplied X clients. It describes the mwm window manager and brief overviews of the various features of a Motif application and is only superficially different from the previous edition. As such does not deserve the title of an OSF/Motif Edition.

Unlike Volume 4 in this series, there is no incentive to overcome the poor typography, as the information that this book contains is not too hard to find elsewhere. This means that this book is not going to serve its intended audience, which is the novice X user.

The other volumes in this series are indispensable to the X Window System user. A much sharper presentation and a much more thorough treatment of Motif is required, if this book is to achieve the same status.

Not recommended.

Book Review

MH & xmh, E-mail for Users and Programmers

Jerry Peek
O'Reilly & Associates, Jan 1991, ISBN 0-937175-63-3. (UK) Price Not yet Available, Soft Back, 555 pp.

"The Octopus Book"

Reviewed by Andrew Macpherson.

Email A.Macpherson@sakura.uucp

This is a book about electronic mail, or rather about MH which is an extremely rich an interface to electronic mail, and its partner XMH which gives a graphic front end to that functionality. How does one separate the book about the program from the program itself? When I started with the Octopus Book I had not used MH at all before, and the distinction was blurred, I could claim to be a serious user of electronic mail systems, but that was about it. I even had to compile up the latest release of MH to be in sync with the text.

Some months on (yes this review is late) the split is much easier to make. MH is a complex system, and the book sets out to present two ways of using it, the style is a friendly, informal tutorial and very readable. The sheer volume of the book is daunting but one swiftly realises that there are really two books and a fairly extensive manual set between the covers. Chapters 4 through 7 describe the command line interface, while 13 to 15 have XMH covered in 88 pages. Much relieved one can move on to read the half of immediate interest.

The book's practical approach gets one up out of the armchair immediately. It may be possible to read, and believe 'this is what happens' in the text, it is more satisfying by far to work through the examples on ones' own system.

Starting with the basic interface one reads, creates, sends and files messages. Then comes the fun of customisation \(em how do you like to list the contents of a mailbox? Do you want to include the original message automaticly in a reply? All the features needed to function as a mail user are well covered. In XMH adding accelerator buttons to print the displayed message is easily done.

And yet... at the end of the section one has had a good tutorial on using mail but one has read a lot of pages without ever touching on auto-filing, "I'm on holiday" messages, bulletin boards or anything else of what one would term the 'power-features' of MH. That first time through one was disappointed by the carefully crafted focus.

A week later I was rude enough to send an automatic "Thanks I'll get round to your message" note to Byron Rakitzis when he replied to a question I had asked about his 'rc' shell. Properly taken to task I turned back to the Octopus Book to see how to do something about it. The information was there, and that is a mistake I will not be making again. My view was changed, and I now appreciate that I had built up the wrong expectations while installing the programs. Read the book first, and borrow someone else's system to work through the examples.

So who should buy this book? A pre-requisite is of course that one will be using or maintaining MH. That in turn begs the question of whether one should be.

Since ucbmail / mailx are rather fundamentally flawed in address handling, something else is needed, and MH is one of the few mail systems that really does obey the standards in every detail. For the mailtool user XMH is a wonderful alternative, and the various emacs modes are great. On the command line though MH is ultimately so rich and flexible that anyone trying to cope with it probably needs this book to get going. I have now learned MH, and I shall continue using it, even running it on my Xenix PC, but did I really want to go that far?

MANAGEMENT COMMITTEE

MINUTES OF MEETING, 5 AUGUST 1991

Present: Pat Duffy, Michael Tuke, Chris Maltby (part of meeting), Frank Crawford, Andrew Gollan, Peter Karr (part of meeting), Scott Merrilees, Rolf Jester.

Meeting commenced at 10:15am.

Also present at the relevant times were ACMS principal Wael Foda and Ellen Gubbin of Symmetry Design.

1 APOLOGIES

Glenn Huxtable.

2 MINUTES OF LAST MEETING (28 JUNE 1991)

Correction: 14.1 Robert Elz - name misspelled.
Moved (PD/MT) that the minutes be accepted. Carried.

3 BUSINESS ARISING FROM THE MINUTES

3.1 Re 3.1: Glenn Huxtable will write a letter thanking the Summer organisers.

Action: GH

3.2 Re 10.2: Peter Karr will submit the AARNET article for /osr to Chris Maltby for review and editing.

Action: PK
and: CM

3.3 Re 10.3: Chris Maltby and Frank Crawford will prepare an information sheet - "How to get on the network."

Action: CM
and: FC

3.4 Re 10.4: We will sponsor an AARNET connection for the Sun User Group in Melbourne.

Action: MT

3.5 Re 14.1: Rolf Jester will follow up with Robert Elz the registration of new business name and the lodgement of the Constitutional changes with Corporate Affairs.

Action: RJ

4 PRESIDENT'S REPORT

Pat Duffy reported:

4.1 The agreed membership renewal changes have been implemented by ACMS. A letter from Pat Duffy describing the changes has gone out to all members and the first billing cycle (1 July) will start within a few days. All unfinancial members will be included in this current billing cycle

4.2 Publicity for AUUG has led to numerous enquiries by phone and mail. Pat has also been invited to speak on "How Open is Open" at a meeting of the Institute of Systems Analysts.

4.3 We should indicate in AUUGN the fact that members can order publications from UniForum, UseNIX and EUUG through AUUG. Chris Maltby and Frank Crawford will contribute a page on this for AUUGN.

Action: CM
and: FC

4.4 We should also order some stocks of likely popular publications for sale at the Conference. Rolf Jester will request ACMS to order 30 each of UseNIX Winter proceedings, UseNIX Summer proceedings and UniForum Products Directory, plus a supply of UniForum membership application forms.

Action: RJ

4.5 Moved (PD/SM) that we record our thanks to Stephen Prince for his substantial contribution to the work of the Committee. Carried unanimously.

Moved (AG/FC) that the President's report be accepted. Carried.

5 SECRETARY'S REPORT

Rolf Jester reported:

5.1 Membership has grown by 45 since the end of June. Membership report and breakdown by State are attached.

5.2 Rolf Jester will ask Ellen Gubbin of Symmetry to quote on new business cards for all Committee Members and the AUUGN Editor, using the new logo style and colour.

Action: RJ

Moved (FC/MT) that the Secretary's Report be accepted. Carried.

6 TREASURER'S REPORT

Frank Crawford reported:

6.1 Balance sheet as at 30/7/91 and Income & Expense accounts for the year to 30/7/91 are attached. There are some final entries required before we have final accounts for the year. Michael Tuke is obtaining the final reports from the accountants.

6.2 Cheque signatories have now been finalised.

6.3 We shall maintain sufficient funds in our cheque account to cover planned expenses and transfer the balance to a cash management account. The Chase AMP term deposit will be re-invested in another interest-bearing deposit.

Action: FC

6.4 Michael Tuke will ask our accountants, Nicol & Nicol for advice on what steps we have to take to ensure that we are tax-exempt.

Action: MT

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

7 AUUGN EDITOR'S REPORT

Frank Crawford reported on behalf of Jagoda Crawford.

7.1 Vol.12 issue 1 is now out. The next issue is almost ready and will be distributed before the end of the month. We still lack:

President's report — Action: PD
Minutes — Action: RJ
Returning Officer's report - attached.

7.2 The third issue for this year will be around the Conference, and the final one in December.

7.3 We shall no longer print the price of back issues in AUUGN so that the back-issue price is not treated by Australia Post as the retail price of AUUGN, which could otherwise result in a higher postage fee.

Action: JC

7.4 Frank Crawford will follow up reduced postage fees with Australia Post.

Action: FC

7.5 We shall continue to print 20 spare copies of each issue for back-issue requests.

Action: JC

7.6 We shall continue to print the list of Institutional members, but try to fit them on to two pages by using smaller type. After the reminders have gone out, we shall drop the names of any unfinancial Institutional members.

Action: JC

7.7 The next issue of AUUGN will be sent to unfinancial members as part of the inducement to renew.

Action: JC

Moved (AG/SM) that the AUUGN Editor's report be accepted. Carried.

8 RETURNING OFFICER'S REPORT

Attached.

9 AUUG'91 Conference

9.1 Andrew Gollan suggested that as part of the post-Conference meeting, we review the process and work-load associated with organising the Conference program.

9.2 Wael Foda will send out the Speakers kits tomorrow (3/8/91), after receiving mailing labels from Andrew Gollan and final contents from Ellen Gubbin.

Action: WF

9.3 Rolf Jester indicated that Conference speaker Mark Shand has not yet been able to obtain funding for his trip from Digital. Rolf will

try to arrange that for him. However it was decided that AUUG would be prepared to pay economy fare if necessary [Rolf Jester did not take part in that decision to avoid conflict of interest.]

Action: RJ
Action: AG

9.4 It was decided that we would pay economy fare for Conference speaker Steve Ruegnitz if necessary. Pat Duffy will negotiate this.

Action: PD

9.5 Peter Karr expressed an interest in publishing selected papers from the Conference in Open Systems Review. Since the copyright for individual papers remains with the author, CMP can agree it directly with the authors.

10 PUBLICITY

10.1 The quote from Symmetry of $8135 for the AUUG exhibition stand at AUUG'91 was accepted. The signs and fittings are re-usable and will thus be paid by AUUG (not by ACMS from the AUUG'91 account). Ellen Gubbin will proceed to have the stand built.

Action: EG

10.2 Ellen Gubbin will quote on member card, certificate and membership brochure. The quote will go to Pat Duffy and be circulated to the Committee as quickly as possible so that we can have all three items in time for the Conference. The design will be faxed for approval by all Committee members.

Action: EG

10.3 The membership card is to identify members. It could be a laminated card with the member details printed on an adhesive label that is affixed to the card before being laminated. Wael Foda will enquire as to cost of a laminating machine. As a simpler alternative, we could use an inexpensive flexible plastic card on which the member details can be printed directly.

The card will be issued to personal members and to the two nominated representatives of Institutional members. It will carry the AUUG logo and the word "Member". It will show member number (personal or institutional), personal name or (Institutional) organisation name, plus the expiry date.

10.4 The certificate is for Institutional members, intended for framing.

10.5 The membership brochure is designed to aid recruiting. It will state AUUG objectives and membership benefits. It will not include

fee amounts or names of officers so as to remain useful beyond the current year. An application form with the fee amount will accompany the brochure. Qty: 5000.

10.6 It was decided that all fifteen invited Conference speakers will be awarded a gift of a high-quality crock of distinctive Australian fortified wine. If possible the crock or the box will be appropriately labelled. The forty other speakers could be awarded the same wine in a normal bottle. Andrew Gollan will make the selection.

Action: AG
and: EG

10.7 The gift for Conference delegates will be a T-shirt. The design, to be chosen by Andrew Gollan with Ellen Gubbin, should highlight AUUG, the event and (most importantly) something to do with UNIX and/or open systems. A Patrick Cook cartoon was suggested.

Action: AG
and: EG

11 OTHER BUSINESS

11.1 It was decided that we would purchase an appropriate modem for ACMS so that we can exchange electronic mail with the Secretariat. Andrew Gollan will select the modem.

Action: AG

11.2 ACSnet enquiries coming to ACMS will be referred to Frank Crawford.

11.3 Rolf Jester will obtain current copy of the Constitution, and will provide a copy to Ellen Gubbin, or at least provide her with a statement of objectives and membership benefits for the brochure.

Action: RJ

12 NEXT MEETING

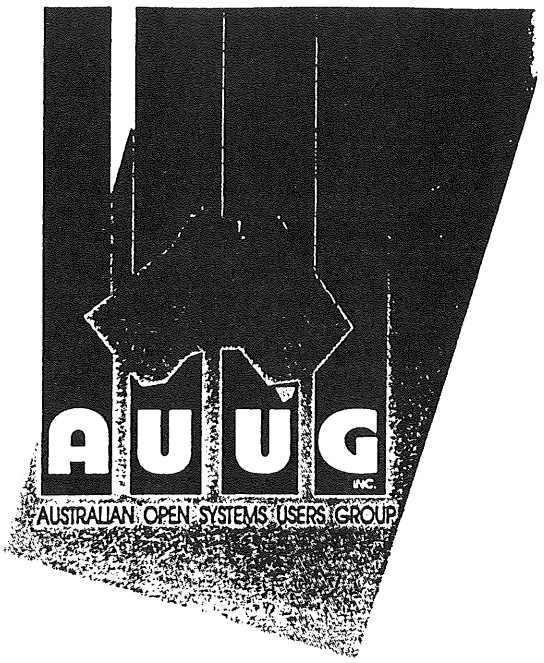
Tuesday, 24 September 1991, 2:00 pm, in one of the conference rooms upstairs above the exhibition hall.

Action: RJ

Meeting closed at 1:30pm

Membership report as at 01/08/1991 **New since last meeting**

Category	*Financial*	*Unfinancial*	*Total*
Institutional	165	62	227
Members	227	151	378
Students	6	4	10
Life Member	1		1
Subscriptions	7	14	21
TOTAL	**406**	**231**	**637**



Membership report as at 01/08/1991

Category	*Financial*	*Unfinancial*	*Total*
Institutional			
ACT	5	3	
NSW	74	25	
NT			
QLD	11	11	
SA	6	4	
TAS	3	2	
VIC	61	11	
WA	5	6	
TOTAL INST.	**165**	**62**	**227**
Members			
ACT	12	7	
NSW	85	43	
NT	1	2	
QLD	19	17	
SA	13	7	
TAS	4	5	
VIC	80	59	
WA	13	9	
O/Seas		2	
Total MEMB.	**227**	**151**	**378**
Students	6	4	10
Life Member	1		1
Subscriptions	7	14	21
GRAND TOTAL	***633***	***382***	***1015***

Balance Sheet for AUUG Inc. 1991/1992

Asset Accounts Date Last Entry: 30/7/91

Asset Accounts	Beginning Balance	Ending Balance	Annual Change	
			Decrease	Increase
Commonwealth Bank	118,421.19	133,523.99	-	15,102.80
Chase AMP	6,000.00	6,000.00	0.00	-
CBA - Int. Term	31,141.59	31,141.59	0.00	-
Total Assets	155,562.78	170,665.58	-	15,102.80

Liability Accounts

Liability Accounts	Beginning Balance	Ending Balance	Annual Change	
			Decrease	Increase
Total Liabilities	0.00	0.00	0.00	-
[Net Worth]	155,562.78	170,665.58	-	15,102.80

Income Statement for AUUG Inc. 1991/1992
Income Accounts Date Last Entry: 30/7/91

Income Accounts	Annual Total		Budget Comparison	
	Budgets	Actuals	Above	Below
Subs-Members	0.00	1,482.00	1,482.00	-
Subs-Institutional	0.00	10,820.00	10,820.00	-
Subs-Newletter	0.00	135.00	135.00	-
Subs-Students	0.00	45.00	45.00	-
Subs-AARNET	0.00	5,095.00	5,095.00	-
Sale-Mail List	0.00	582.50	582.50	-
Sale-Video	0.00	55.00	55.00	-
AUUG 91 Conf & Exh	0.00	0.00	0.00	-
Uniforum - Rebate	0.00	0.00	0.00	-
Uniforum - Refund	0.00	1,654.48	1,654.48	-
Check Interest	0.00	0.00	0.00	-
Other Income	0.00	14.00	14.00	-
Total Income	0.00	19,882.98	19,882.98	-

Expense Accounts

Expense Accounts	Annual Total		Budget Comparison	
	Budgets	Actuals	Above	Below
Check Charges	0.00	15.37	15.37	-
Secretariat fees	400.00	200.00	-	200.00
Accountancy fees	0.00	0.00	0.00	-
Advertising	0.00	0.00	0.00	-
Filing fees	0.00	0.00	0.00	-
Freight & Cartage	0.00	0.00	0.00	-
General expenses	0.00	0.00	0.00	-
Hire of equipment	0.00	0.00	0.00	-
Insurance	0.00	0.00	0.00	-
Printing & Stat.	0.00	1,760.00	1,760.00	-
Postage	0.00	79.58	79.58	-
Subscriptions	0.00	0.00	0.00	-
Committee Expenses	0.00	45.00	45.00	-
Telephone	0.00	0.00	0.00	-
Travelling Expense	0.00	0.00	0.00	-
Refund - Memb fees	0.00	325.00	325.00	-
AARNET Expenses	0.00	125.00	125.00	-
Membership Adminis	0.00	730.23	730.23	-
Marketing	0.00	1,500.00	1,500.00	-
Total Expense	400.00	4,780.18	4,380.18	-
[Net Income]	(400.00)	15,102.80	15,502.80	-

John O'Brien, the AUUG Returning Officer, has asked me to post this article announcing the result of the Elections for the positions of General Committee member and also the approval of Affiliation with UniForum.

Positions previously filled (unopposed):

President:	Pat Duffy	(pat.duffy@amail.amdahl.com)
Vice-President:	Chris Maltby	(chris@softway.sw.oz.au)
Secretary:	Rolf Jester	(rolf.jester@sno.mts.dec.com)
Treasurer:	Frank Crawford	(frank@photon.lhrl.oz.au)

The successful cadidates were:

General	Andrew Gollan	(adjg@softway.sw.oz.au)
Committee	Glenn Huxtable	(glenn@cs.uwa.oz.au)
Members:	Peter Karr	(??)
	Micheal Tuke	(mjt@anl.oz.au)
	Scott Merilees	(Sm@bhpese.oz.au)

The new AUUGN editor is:

	Jagoda Crawford	(jc@atom.lhrl.oz.au)

Public Officer:	Robert Elz	(kre@munnari.cs.mu.oz.au)
Returning Officer:	John O'Brien	(john@wsa.oz.au)

AUUG Secretariat: (02) 361 5994

Affiliation with UniForum: Carried Yes: 72 No: 2

I would like to take this opportunity to thank Peter Barnes for his efforts as Secretary, during which time AUUG finally adopted a professional membership service. As a result of this work members can expect a much higher level of membership service.

Thanks are also due to David Purdue the retiring AUUGN editor, who did a great job for 18 months until pressures of work produced the recent hiatus. The editor's job is always made difficult by a low rate of contributions from the membership, and is now being made even more so by Australia Post's silly rules for registered publications.

Michael Tuke has retired from the Treasurer's post but has been returned to the Committee. Thanks are due to Michael for bringing a new standard of financial professionalism to AUUG. We look forward to his continued participation on the Committee.

Finally (but not least) to Stephen Prince for his service as a Committee member. Stephen certainly did more than his fair share of the Committee's work.

AUUG is aiming for continued improvement in the standard of all our activities, especially for AUUG91 at Darling Harbour, 25-27th September. Members and others on our mailing list will receive details soon, or call the secretariat for more information.

Chris Maltby
Vice President

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:

Tuesday, 1 October 1991
Wednesday, 13 November 1991
Thursay, 12 December 1991
Tuesday, 21 January 1992
Wednesday, 4 March 1992
Thursay, 16 April 1992
Tuesday, 26 May 1992
Wednesday, 8 July 1992
Thursay, 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 newsgroup **aus.auug**.

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 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
P O Box 366
Kensington NSW 2033
Australia

- Please don't send purchase orders — perhaps your purchasing department will consider this form to be an invoice.
- Foreign applicants please send a bank draft drawn on an Australian bank, or credit card authorisation, and remember to select either surface or air mail.
- Use multiple copies of this form if copies of AUUGN are to be dispatched to differing addresses.

This form is valid only until 31st May, 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
P O 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: .....

.....

.....

.....

---

ffice use only:

ate: ___/___/___

ho: _____ *Memb#* _____

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
P O 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 usally 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.

- ☐ System V.3 source
- ☐ System V.2 source
- ☐ System V source
- ☐ System III source
- ☐ 4.2 or 4.3 BSD source
- ☐ 4.1 BSD source
- ☐ V7 source
- ☐ System V.3 binary
- ☐ System V.2 binary
- ☐ System V binary
- ☐ System III binary
- ☐ Other *(Indicate which)* .....

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
P O Box 366
Kensington NSW 2033
Australia

- **Please don't send purchase orders — perhaps your purchasing department will consider this form to be an invoice.**
- **Foreign applicants please send a bank draft drawn on an Australian bank, or credit card authorisation, and remember to select either surface or air mail.**

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: _____