AUUGN Volume 11, Number 3 June 1990

Australian UNIX systems User Group Newsletter

AUUGN

Volume 11, Number 3

June 1990

Registered by Australia Post, Publication Number NBG6524

The Australian UNIX* systems User Group Newsletter

Volume 11 Number 3

June 1990

CONTENTS

AUUGN is the journal of the Australian UNIX[1] systems User Group (AUUG Incorporated).

[1] UNIX is a registered trademark of AT&T in the USA and other countries.

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

AUUG Executive

President	**Greg Rose** *greg@softway.sw.oz.au* Softway Pty Ltd New South Wales	Vice President	**Pat Duffy** *pzd30@juts.ccc.amdahl.com* Amdahl Australia Pty Ltd New South Wales
Secretary	**Peter Barnes** *pdb@uqcspe.cs.uq.oz.au* Computer Science University of Queensland	Treasurer	**Michael Tuke** *mjt@anl.oz.au* ANL Limited Victoria
Committee Members	**Frank Crawford** *frank@teti.qhtours.oz.au* Q.H. Tours Pty Ltd New South Wales		**Andrew Gollan** *adjg@softway.sw.oz.au* Softway Pty Ltd New South Wales
	Chris Maltby *chris@softway.sw.oz.au* Softway Pty Ltd New South Wales		**Scott Merrilees** *sm@bhpese.oz.au* BHP Information Technology New South Wales
	Stephen Prince *sp@labtam.labtam.oz.au* Chancery Lane Computer Services Pty Ltd Victoria		

Next AUUG Meeting

The AUUG'91 Conference and Exhibition will be held from the 24th to the 27th of September, 1991, at Darling Harbour, Sydney. The AGM of AUUG Inc. will be held during the conference.

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

Well, it has been some time.

Despite the date shown on the cover this issue is coming out after the AUUG'90 Conference and Exhibition, which was a huge success. Over 400 people attended the conference, over 170 people took part in the tutorial program and over 1200 people walked through the exhibition (not counting conference attendees). There was an interesting mix of papers with commercial content and those of technical interest, and the World Congress Centre is the best conference venue AUUG has ever used. I look forward to meeting you all again at AUUG'91 in Sydney next September.

The AUUG Annual General Meeting was held during the conference, and members took the opportunity to air their views to the Management Committee. One issue raised was that the AUUG could act as a coordinator for members trying to get a connection to the network. This issue is being followed up, and a survey form appears in this issue. Please take the time to fill it out and send it in if you are seeking a net connection or, more importantly, you can offer one. Reports and minutes from the AGM should appear in the next issue of AUUGN.

Following the success of the Summer Technical Meetings in February this year, the AUUG Management Committee is hoping to stage them again in '91. Many thanks must go to Glenn Huxtable who has kindly (bravely? foolishly?) volunteered to act as national coordinator again. However, we are still looking for organisers in each state, and of course for people to present technical papers.

In this issue there is another book offer from Prentice-Hall. Prentice-Hall kindly offer AUUG members a 20% discount on these books, but there is a catch - they want to see book reviews published in AUUGN. Now, while I have had little trouble finding reviewers, I have had a lot of trouble getting reviews out of them. Prentice-Hall donate review copies of the books to us, so please don't volunteer to review a book unless you are prepared to devote the time to read the book and write a review and get it back to me within a month to six weeks. I am starting to get an idea of who the reliable book reviewers are, and these people will get preference for titles as they come in.

That's it from me. Keep those cards and letters rolling in.

AUUGN Correspondence

All correspondence regarding the AUUGN should be addressed to:-

David Purdue
AUUGN Editor
PO Box 366
Kensington, NSW, 2033
AUSTRALIA

ACSnet:	auugn@munnari.oz
Phone:	+61 3 353 3913 (w)
	+61 3 813 1258 (h)
Fax:	+61 3 353 2987

Contributions

This Newsletter is published approximately every two months. The deadline for contributions for the next issue is Friday the 26th of October 1990.

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

I prefer documents to be e-mailed to me, or mailed to me on a floppy disk (IBM-PC 5-1/4 inch or 720K 3-1/2 inch; or Macintosh 3-1/2 inch), and in plain text format. 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.

AUUG Newsletter

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 AUUG 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, details are printed at the end of this issue.

Acknowledgements

This Newsletter was produced with the kind assistance of and on equipment provided by the Advanced Imaging Systems department of Kodak (Australasia) Pty Ltd. I would also like to thank Labtam Information Systems Pty Ltd for providing me with a network connection.

Disclaimer

AUUG Institutional Members

ACUS - UNISYS
ANL Limited
Adept Business Systems Pty Ltd
Alcatel STC Australia
Aldetec Pty Ltd
Alliance Computer Centre Pty Ltd
Apple Computer Australia
Apscore International Pty Ltd
Australian Artificial Intelligence Institute
Australian Electoral Commision
Australian National University
Australian Nuclear Science & Technology Organisation
Australian Wool Corporation
BHP Melbourne Research Labs
Ballarat Base Hospital
Basser Department of Computer Science
Bond University Library Service
Bond University School of Information and Computing Science
Bureau of Meteorology
Bureau of Vocational, Further Education and Training
CADAD Support Section - SECWA
Capricorn Coal Management Pty Ltd
Civil Aviation Authority
Co-Cam Computer Group
Colonial Mutual
Commodore Business Machines Pty Ltd
Commonwealth Department of Primary Industries and Energy
Comperex (NSW) Pty Ltd
Computer Power IR+D, NSW Branch
Computer Power Today IR+D
Computer Software Packages
Corinthian Engineering Pty Ltd
Crane Enfield Metals Pty Ltd
Cybergraphic Systems Pty Ltd
DBA Limited
DMR Group
Data General
Davey Products Pty Ltd
Deakin University
Department of Industrial Relations & Employment
Department of Transport, Queensland
Dept of Agricultural & Rural Affairs
Dept of Industry, Technology and Resources, Victoria
Digital Equipment Corporation (Australia) Pty Ltd
ERIN, Bureau of Flora and Fauna
Earth Resource Mapping Pty Ltd
Elxsi Australia Ltd
Epson Australia Pty Ltd
Exicom Australia Pty Ltd
Flinders University - Discipline of Computer Science
Fremantle Port Authority
Geelong and District Water Board
Genasys II Pty Ltd
Golden Casket Art Union
Gould Electronics Pty Ltd
Hamersley Iron Pty Ltd
Harris & Sutherland Pty Ltd
Hewlett Packard Australia Limited
Hewlett-Packard Australian Software Operation
Honeywell Software Centre
IBM Australia Ltd
ICL Australia Pty Ltd

AUUG Institutional Members

IPS Radio and Space Services
Ipec Transport Group
Kodak (Australasia) Pty Ltd
Labtam Information Systems Pty Ltd
Logic Group
Macquarie Bank Limited
Macquarie University
Mathematics and Computing Department - BCAE (Kelvin Grove Campus)
Mincom Pty Ltd
Motorola Communications Australia
NEC Information Systems Australia Pty Ltd
NSW Parliament
National Engineering Information Services Pty Ltd
Nixdorf Computer Pty Limited
OPSM
Olivetti Australia Pty Ltd
Olympic Amusements Pty Ltd
Overseas Telecommunications Corporation
Pact International
Port of Melbourne Authority
Prentice Computer Centre
Prime Computer of Australia Ltd
Pyramid Technology Corporation
Q. H. Tours Limited
Queensland Department of Mines
Queensland Education Department
Queensland Justice Department
RMIT Computer Centre
Roads and Traffic Authority NSW
SEQEB
Sigma Data Corporation Pty Ltd
Silicon Graphics Computer Systems
Sola International Holdings Ltd
South Australian Institute of Technology
Sphere Systems Pty Ltd
Stallion Technologies Pty Ltd
Stamp Duties Office Victoria
State Bank Victoria
State Bank of NSW
State Library of Tasmania
Steedman Science and Engineering
Sugar Research Institute
Sun Microsystems Australia
Swinburne Institute of Technology
Sybase Australia Pty Ltd
Tandem Computers Pty Ltd
Tasmania Bank
Tattersall Sweep Consultation
Tech Pacific
Telecom Business Services
Telecom Network Engineering - CSS
University College of Central Queensland
University of Adelaide
University of Melbourne (Information Technology Services)
University of New England
University of New South Wales
University of Technology Sydney - Computing Services Division
University of Wollongong
Vicomp
Wacher Pty Ltd
Wang Australia Pty Ltd
Wyse Technology Pty Ltd
Yartout Pty Ltd

Letters to the Editor

From Tim Roper,
Labtam Information Systems Pty Ltd

Dear Sir,

Thank you for publishing the paper *Design of the Labtam Xengine* in AUUGN 11(2) as presented at the AUUG Summer '90 meeting. I refer to the *xbench* figures published on Page 42. At the time the paper was presented in February they were already overtaken and the rate at which they have improved since then means that they are seriously out of date at the time of (belated) publication. For example, the overall xstone rating of both the CT100 and MT200 has nearly doubled. In the interests of accuracy I would like to take this opportunity to tender the following updated summary and would be grateful if you would publish this letter in the next issue of AUUGN.

machine	planes	comm	line	fill	blt	text	arc	cmplx	xstones
MT200	1	10Mb ether	43202	27357	57484	86968	1266467	41764	49182
CT100	8	10Mb ether	79852	22959	20709	110343	1210344	28692	39732
Sun3/50 (R3)	1	unix-socket	10000	10000	10000	10000	10000	10000	10000

Attached is the complete *xbench* output should you find space for it. Readers who would like an updated hardcopy of the paper, or more extensive *xbench* or *x11perf* results, are welcome to contact

Technical Publishing Department
Labtam Information Systems Pty Ltd
41 Malcolm Road
Braeside VIC 3195

Phone: (03) 587 1444
Fax: (03) 580 5581

or e-mail their name and postal address to **xengine@labtam.oz.**

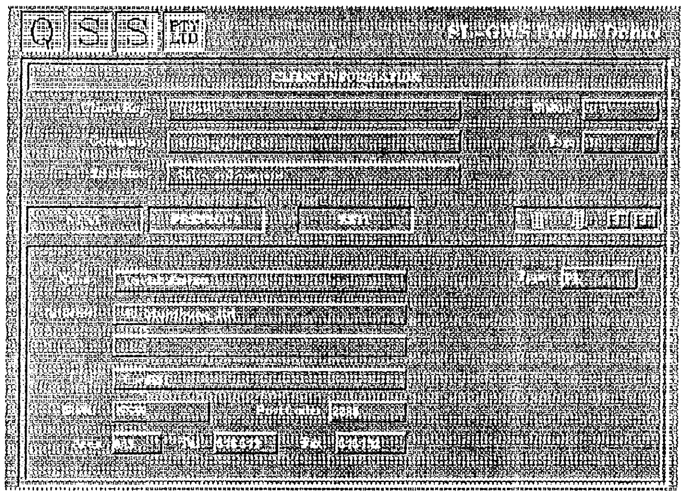
Yours sincerely,

Timothy Roper
Senior Systems Programmer

[I choose not to reproduce the lengthy xbench results, they can be obtained from the above source. I know Labtam are still optimising the Xengine server, so these figures may be out of date already by the time this gets published - Ed.]

NOW

Beyond Widgets



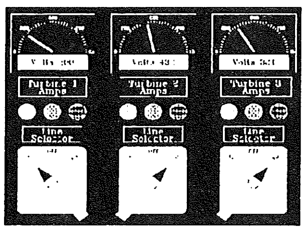
SL-GMS...the complete tool for application graphics.

The Graphical Modelling System from SL Corporation is the only complete application graphics tool for UNIX and VMS workstations, including 386-based UNIX platforms.

Draw new graphical objects · Connect easily to data sources
Animate to visualise real-time data · Use to control the application

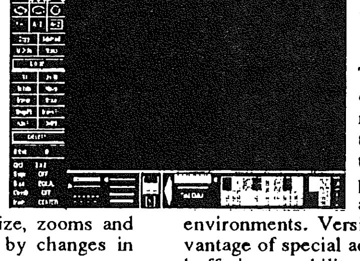
Standard widgets are not enough:

With the rise of graphics workstations has come a demand for tools that speed the development of graphics screens for applications. A bewildering array of tools has appeared to aid developers with X Windows and primary GUI styles: MOTIF, Open Look and DECwindows. Many of these tools are WYSIWYG editors limited to the creation of standard widgets such as menus, scroll boxes, sliders and buttons. Standard widgets, however, are not enough for application visualisation. inevitably, the need arises for custom screen objects (graphs, maps, icons and other pictures) which are beyond such tools, and which are too time-consuming to create with Xlib. Developers also need a way to visualise changing data in real time.



A complete graphics tool must provide:

A Powerful Drawing Tool - The ability to create custom screen objects. Not limited to canned graph types, this flexible tool with many CAD features and an interface like familiar PC drawing tools allows you to draw whatever you need, attach dynamic behaviours and test those behaviours-all within the drawing tool.



Dynamics - The ability of screen objects to instantly reflect changes in data values, receive user input or execute callback functions. Over forty attributes such as colour or text changes, visibility on/off, percent fill, line width, rotation, movement, scaling, font style or size, zooms and window creation can be triggered by changes in data values or user input.

Complete Xt Widget Integration - SL-GMS graphics can fully incorporate Xt widgets. Screen objects created with SL-GMS fully interact with MOTIF, Open Look, DECwindows or other toolkit widgets, whether in the same or different windows.

GISMOs - Graphical Interactive Screen Management Objects are called GISMOs to distinguish them from Xt widgets. Fully interactive with Xt widgets, GISMOs can take any appearance you wish and trigger any user-defined function or external program. Created with the drawing tool, GISMOs provide developers with tremendous design flexibility.

HyperCard-like Screen Management - After screens have been created, the user must be able to button from any screen to any other screen in the application. With the screen management System (SMS) included in SL-GMS, the developer can give the user this ability without writing a line of code. SMS can bring up new screens with data sources attached and dynamics up and running.

Data Source Management - The ability to connect screen objects to data sources such as files, databases, expert systems and real-time feeds.With the Data Source Manager of SL-GMS, these connections are easily made.

Runtime Editors and Configurators - The ability of the enduser to customise or reconfigure screens to match the current environment.

Cross-Platform Portability - The ability to develop screens on any major workstation and run them on any other through simple ASCII file transfer. SL-GMS also supports PixWin, Iris GL, GKS and other non-X graphics environments. Versions such as Iris GL take advantage of special accelerator hardware and double buffering capability.

SL-GMS is widely used:

For real-time or highly interactive applications in fields such as manufacturing, process control, network management, cockpit display and financial trading.

Licensees of SL-GMS include:
ABB/Combustion Engineering
Applied Automation, Inc.
The Boeing Company
Bell Canada
Chrysler Motor Corporation
Cummins Engine Company
Eurotherm Ltd.
General Electric Company
Hewlett-Packard
Hughes Aircraft Company
Johns Hopkins University, APL
Lawrence Livermore National Labs
Lockheed
McDonnell-Douglas
Martin Marietta
Nothrop Corporation
Peugeot
Westinghouse Nuclear Division
In Australia:
BHP Steel
Toshiba Power Division
Digital Equipment Corporation

Supported Workstations:

Sun (PixWin or X), DEC (VMS or ULTRIX), Silicon Graphics (X or GL), HP, Apollo, IBM, and MIPS as well as 386-based workstations running UNIX and the X windowing System.

Q.S.S. Pty. Ltd.
40 Munibung Rd., Cardiff NSW 2285.
PO Box 269, New Lambton 2305.
Tel. (049) 546524 Fax. (049) 545132

Q S S PTY LTD

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:* *E-Mail:*	(02) 332 4066 auug@teti.qhtours.oz

Technical enquiries to:

Frank Crawford	(frank@teti.qhtours.oz.au)	(02) 957 0348
or		
Scott Merrilees	(Sm@bhpese.oz.au)	(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	MHSnet SLIP specify):	UUCP ________________
A6.	Type of connection:	Direct X.25/Dialin Other (Please	Modem/Dialin X.25/Dialout specify):	Modem/Dialout ________________
A7.	If **modem**, connection type:	V21 (300 baud) V22bis (2400) Other (Please	V23 (1200/75) V32 (9600) specify):	V22 (1200) Trailblazer ________________
A8.	Estimated traffic volume (in KB/day): (not counting netnews)	< 1 > 100:	1-10 estimated volume:	10-100 __________
A9.	Do you require a news feed?	Yes Limited (Please	No specify):	 ________________
A10.	Any time restrictions on connection?	Please specify:	________________	
A11.	If the connection requires STD charges (or equivalent) is this acceptable?	Yes	No	
A12.	Are you willing to pay for a connection (other than Telecom charges)? 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:			

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 the South Eastern Suburbs. The aims of SESSPOOLE are:

> To promote knowledge and understanding of Open Systems; and to promote knowledge and understanding of Open Bottles.

(Note that these aims have been updated in line with recent changes to the aims of AUUG Inc.)

SESSPOOLE was the first Chapter of AUUG Inc to be formed, and members of SESSPOOLE were involved in the staging of the AUUG Summer'90 meeting.

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:

Thursday, 15th November, 1990
Tuesday, 15th January, 1991
Wednesday, 27th February, 1991
Thursday, 18th April, 1991
Tuesday, 28th May, 1991
Wednesday, 17th July, 1991
Thursday, 29th August, 1991

Hope we'll see you there!

For more information on SESSPOOLE and SESSPOOLE activities (including a description of how much fun it is to book a table in a restaurant under the name "SESSPOOLE"), contact either David Purdue (ph. (03) 353 3913, e–mail: *auugn@munnari.oz.au*) or John Carey (ph. (03) 587 1444, e–mail: *john@labtam.labtam.oz.au*), or keep a lookout for announcements in **aus.auug**.

CONVENTION UNIX 91

4th Conferences Series and Exhibition

Preliminary Announcement and Call for Papers

The French UNIX and Open Systems Users Group (AFUU), in cooperation with the Bureau International des Relations Publiques (BIRP), is organising the Convention UNIX 91. This event is structured around a series of conferences, tutorials and a commercial exhibition of hardware and software dedicated to UNIX and Open Systems. The Convention will be held from the 26th to the 30th of March 1991 at the CNIT, Paris La Defense.

Being open and international, the Conferences Programme Committee is setting up three main streams of information to enable users, researchers and manufacturers to create for themselves a wide forum for exchange and dialogue:

Technical Conferences **Users Sessions** **Industrial Solutions**

Technical Conferences

These sessions will give the opportunity to familiarise with the latest trends in research and development. The Programme Committee suggests the following subject areas:

- Distributed Systems and Cooperative Processing
- Networks: Interconnections, Interoperability and Management
- System Administration and Security
- New Technologies for Open Systems
- Parallelism and New Architectures
- Graphics Applications
- Databases
- Object Oriented Applications and Interfaces
- Development and Runtime Environments

Users Sessions

Today, more than ever, UNIX and the Open Systems world is ruled by the users. These sessions are intended to be a forum where these users, coming from different technical areas, can meet and share their experiences and analysis of their hardware and software installations. Two major tracks are proposed for these sessions:

- Users Experiences, with the following topics:
  - configurations (mini and mainframes, microcomputers, networks, ...)
  - standard applications (office automation, management, ...)
  - specific applications (multimedia, industrial processing, ...)
  - various facettes of UNIX insertion (professional, cultural, technical)
  - choice guidelines (benchmarks, validation, certification, ...)

- Open Systems Stakes, with the following topics:
  - market and trends
  - international standards vs. de-facto standards
  - applications portability

Industrial Solutions

In addition to a large exhibition where they can show and demonstrate their products, industrialists, manufactuturers and software vendors will be able to participate in these sessions concerning existing or forthcoming products.

CONVENTION UNIX 91

Suggested topics are:

- Operating System (new developments, real time, system administration tools, tools standardisation, interoperability)
- User Interfaces
- Tools (security, project management, electronic mail)
- General Purpose and Specific Software Packages
- Databases (building tools, access tools, distributed databases, object oriented databases, databases and artificial intelligence)
- Software Engineering
- Networking (mail, VIDEOTEX, ISDN)
- Office Automation and Desktop Publishing

Method of submission

9th November 1990: Deadline for the receipt of communications or extended abstracts by the Conference Secretariat.

7th December 1990: Notification to authors of the papers selected by the Program Committee.

10th January 1991: Deadline for the reception of the full papers by the Conference Secretariat.

Submissions should include: a title, the author's name and affiliation and the target audience (Technical Conference, Users Session, Industrial Solutions). Official languages for the Conferences are French and English. French/English translation services will be provided. Submissions should be sent to the following address:

A.F.U.U.
Secretariat de la Convention 91
11, rue Carnot
94270 Le Kremlin Bicêtre
France

Telephone:	(+33)-1-46.70.95.90
Fax:	(+33)-1-46.58.94.20
Telex:	263887F
E-Mail:	afuuconf@inria.inria.fr

CONVENTION UNIX 91

Programme Committee Chair:

Sylvain Langlois
EDF - Direction des Etudes et Recherches
IMA/ICI/IDR, M-305
1, avenue du General de Gaulle
92141 Clamart CEDEX, France

Telephone:	(+33)-1-47.65.44.02
Fax:	(+33)-1-47.65.35.23
E-mail:	sylvain@cli53an.edf.fr

Conferences Proceedings Preparation:

Philippe Dax
Telecom Paris

E-mail:	dax@enst.fr

"Technical Conferences" Chair:

Jean-Christophe Petithory
Universite Paris VIII
Departement Informatique
2, rue de la Liberte
93526 Saint-Denis CEDEX 2, France

Telephone:	(+33)-1-49.40.64.00
E-mail:	jcp@uparis8.univ-paris8.fr

"Users Sessions" Chair:

Jean-Michel Cornu
Consultant
69, rue de Seine
91130, Ris Orangis, France

Telephone:	(+33)-1-69.43.48.47
Fax:	(+33)-1-60.78.00.98
E-mail:	Jean-Michel.Cornu@enst.fr

"Industrial Solutions" Chair:

Jean-Luc Anthoine
IUT Departement Informatique
Rue Engel Gros - BP 527
90016 Belfort CEDEX, France

Telephone:	(+33)-84.21.01.00
Fax:	(+33)-84.22.29.05
E-mail:	anthoine@afuu.fr

AUUG Book Club

Earlier this year AUUG Incorporated and Prentice Hall Australia formed the AUUG Book Club to give AUUG members a chance to obtain Prentice Hall books at a discount.

To obtain copies of the books reviewed here, fill out the order form that appears on page 22 and send it to Prentice Hall at the address shown, or place a phone order by ringing Liz Guthrie on (02) 939 1333. Don't forget to mention that you are an AUUG member, and to deduct 20% from the Recommended Retail Price shown.

Review copies of books are kindly provided by Prentice Hall.

Books For Review

I am currently seeking people to review the following titles:

Philip E. Bourne:
"UNIX For VMS Users",
ISBN 0-13-947433-1, pp 368, RRP $84.95, 1990

Prabhat K. Andleigh:
"UNIX System Architecture",
ISBN 0-13-949843-5, pp 274, RRP $57.95, 1990

Ulka Rodgers:
"UNIX Database Management Systems",
ISBN 0-13-945593-0, pp 338, RRP $39.95, 1990

Paul Mahler:
"An Informix-4GL Tutorial",
ISBN 0-13-464173-6, pp 282, RRP $43.95, 1990

Nathaniel S. Borenstein:
"Multimedia Applications Development With The Andrew Toolkit",
ISBN 0-13-036633-1, pp 310, RRP $72.95, 1990

The more astute among you may notice that there are many more books mentioned on the order form than there are reviews in the next few pages. That is because most of the people who promised me book reviews did not produce them (to those people - I still want those reviews!). This annoys me, because it means I miss out on copy for AUUGN, and also because it jeopardizes the relationship we have with Prentice Hall (they expect to get reviews out of us, that's why they supply the books).

So while I do not wish to discourage people from volunteering to review books, please realise that I will expect reviewers to produce a review within about a month of receiving the book. People I know to be reliable reviewers will be more likely to get their choice of books to review.

If, after all that, you would like to review one of the books listed here, please phone or e-mail the AUUGN Editor. My numbers and address are shown on page 4 of this issue.

David Purdue, AUUGN Editor

AUUG Book Club

Book Reviews

X/OPEN PORTABILITY GUIDE VOLUME 4 (PROGRAMMING LANGUAGES)

by The X/Open Company Ltd
ISBN 0-13-685868-6, RRP $52.50, 1988

Reviewed by Bryn M. Pears
Telecom Research Laboratories
<b.pears@trl.OZ>

The X/Open portability guide is issued by the X/Open Company, Ltd. Claimed by themselves to be a major breakthrough in Open Systems, X/Open are a group of "major information system suppliers", supported by software developers, etc. Their stated goal is "to make true Open Systems a practical reality". This goal is apparently achieved through common applications environments, etc. Volume 4 of Issue 3 of the X/Open portability guide deals with two programming languages (C and COBOL) and what is necessary to make ones applications conform to the X/Open standards.

Volume four of the guide is divided into two parts.

The first part is the C half. These four chapters deal with X/Open's attitude to C portability.

Chapter one introduces the C section, describing the contents of the various chapters and adding a little detail to the reader's picture of the X/Open philosophy.

Chapter two presents a detailed C language definition. This definition is practically identical to that given in the AT&T UNIX system V programming guide, release 2.0 and as such will be familiar to many readers.

Chapter three was by far the most interesting of the four, dealing with some of the issues involved in the production of portable C code. Advice is given for programmers trying to produce such code. This chapter is organised in a fairly ad hoc fashion, being essentially a list of points to watch for during the coding and testing processes. The ANSI draft standard is also mentioned, with the differences between this standard and the X/Open stance being highlighted.

Chapter four deals with Lint, giving a brief overview of its function, exhorting its use, and detailing how to use it. The meanings of Lint outputs are explained. This chapter provides a useful beginners guide to Lint.

The second half of the guide contains five chapters and deals with X/Open's approach to open systems COBOL. X/Open's COBOL definition is based upon the ANSI standard, ANS X3.23-1985 and its international counterpart I 1989:1985, with a number of extensions. The most notable of these is the screen handler.

Chapter five introduces the X/Open COBOL standard, noting its derivation.

Chapter six presents the full X/Open COBOL definition, consisting of wholesale quotes of IS 1989:1985 plus extensions to cover perceived deficiencies of that standard. This is presented in the form of a syntax summary and is clear and easily read. All X/Open extensions to the COBOL standard are clearly marked as such.

Chapter seven defines the X/Open extensions in a more rigorous fashion, dealing with each in turn. Much attention is given to the X/Open Screen Handling Module, the major extension provided. Other additions include such vital things as the EBCDIC character set.

Chapter eight provides a summary of extensions and limitations. Exactly which sections of ANS X3.23-1985 are excluded from the X/Open standard is stated here, as are all of the extensions to the standard.

Chapter nine addresses the specifics of X/Open compliant systems and the effects of such compliance on currently used COBOL compilers and code. Recommendations are made as to how maximum portability for COBOL code may be achieved.

Overall this book is well presented and laid out. It is clear, informative and easy to read. The problem is that its use seems to be extremely limited unless you happen to be working in an X/Open compliant site or are considering becoming one. I found the discussions of portability issues quite interesting but the bulk of the volume is consumed by language definitions. A moderately interesting read but I wouldn't buy it myself.

AUUG Book Club

Book Reviews

X/OPEN PORTABILITY GUIDE VOLUME 6 (WINDOW MANAGEMENT)

by The X/Open Company Ltd
ISBN 0-13-685884-8, RRP $52.50, 1988

Reviewed by Huw Davies
LaTrobe University Computer Centre
<cchd@latvax8.lat.oz.au>

From the start it should be said that this is not the sort of book that is read from cover to cover, rather it is the type of book that, in the appropriate environment, will be referred to so often that many pages will become very 'dog-eared'. What this volume (one of a series of seven) provides is an accessible reference to the clib bindings for the X windows system and would prove invaluable to an experienced X-windows programmer.

Why this series of books? To quote from the Preface: "*X/Open represents a major breakthrough in the world of Open Systems. A large number of the world's major information system suppliers, supported by representatives of the user, system integrator and software development communities have come together to make true Open Systems a practical reality.*

This is achieved by the establishment of a comprehensive integrated Common Applications Environment which ensures portability and connectivity of applications and allows users to move between systems without retraining.

The X/Open Portability Guide contains an evolving portfolio of practical integrated standards for application portability. All X/Open members guarantee to support the defined interfaces."

The book is divided into five chapters and two appendices. Chapter 1 contains a very brief (one page) overview of the X-windows system; a description of the format of Chapter 2; a good definition of the terminology of the X windows system; a very brief description of the fundamental concepts of the X windows system and how connections are made to an X server; caveats and error processing and a complete list of all the functions and macros that are provided by an X/Open compliant Xlib implementation.

Chapter 2 lists all the functions contained in the Xlib library. Each entry contains the function name; the synopsis (how the function is defined); a description of the function parameters and the function itself; possible error returns and a change history. As would be expected, this chapter occupies the majority of the volume.

Chapter 3, entitled "Event Handling and Predefined Values", describes both how *events* ("data generated asynchronously by the X server as a result of some device activity, or as a side-effect of a request sent by an Xlib function") are generated and describes in detail what processing occurs when event masks are passed to `XSelectInput()`. The second section describes *properties* (the collection of named typed data); the use of colour in X and the key values (the device independent mapping of keys to internal keycap values).

The format of the different files used by X clients is briefly described in Chapter 4. The types described are bitmap and pixmap files, font files and resource files. The final chapter and the following two appendices describe the contents of the four include files which are part of the X windows system; the various X data structures and the macros defined to make access to the data structures more convenient.

From the description of the contents, it can be seen that this book is not a tutorial introduction (there are a number of books which aim to provide this, for example "Introduction To The X Window System" by Oliver Jones), but a reference for an experienced X programmer who wishes to write X/Open compliant programs. Typical usage would be in checking the form of an Xlib call, and ensuring that the application remains compliant. To this end, only Chapter 2 is of interest, but the other chapters provide, in a convenient accessible form, information that is otherwise inconvenient, if not difficult, to come by.

The only drawback with this volume is that the copy reviewed dates from August 1988 and documents X11 Release 2, and whilst still applicable, does not contain information about the new features in X11 Release 3 and the recently announced Release 4.

AUUG Book Club

Book Reviews

X/OPEN PORTABILITY GUIDE VOLUME 7 (NETWORKING SERVICES)

by The X/Open Company Ltd
ISBN 0-13-685892-9, RRP $52.50, 1988

Reviewed by Douglas Ray
University Of Melbourne Information Technology Services
<doug@murtoa.cs.mu.oz>

first impression:

Thin card cover. Cheap wire comb binding designed to warp said cover. No pagination on the table of contents. A scant cm thick, and all for only $50... thank you, Prentice-Hall. How do you do it?

between the covers:

The book documents a library of 25 networking functions. These form the X/Open Transport Interface (XTI), providing access to the transport services; both OSI (connection-oriented and connectionless modes) and Internet (tcp/udp) protocols are supported.

Note that we aren't discussing executable code, here. The book merely proposes a standardised set of C functions to implement the above protocols. 'X/Open' is a commercial consortium whose ostensible purpose is,

> "to adopt and adapt existing standards
> into a consistent environment."

The central part of the book is a series of 'man'-style pages for the 25 networking functions. The only difference from your average section 2 or section 3 entry is a formalised description of parameters, both before and after invocation. Commendable, and we hope in some future form less cryptic.

The remainder of the text, on either side of the 'man' pages, gives a detailed description of the available protocols. This is generally structured well, and is pleasantly concise. There is considerable redundancy between some sections, though, almost to the point of 'padding'.

A thorough explanation of both the protocols and the abstractions these protocols operate on is presented: state and event tables, synchronous and async operation, connection mode (fixed route) and connectionless (datagram) options are clearly documented. The manual tabulates the mandatory and optional functions it defines. What I was looking for, without luck, was a corresponding clear statement of where their "adoption... of existing standards" finished and their "adaption" of them began; however, any mapping of OSI's service functions onto, for example, C functions is beyond the scope of the OSI standards: this is why a document such as this X Transport manual can be useful.

The contents discussed to this point seem quite reasonable. The book warns that it is not self-sufficient as a network transport interface: all the system dependent bits and pieces are subsumed into 'Event Management', which is defined in a separate paper; a user supplied 'Event Management' library would be necessary for synchronous operation. This we find less reasonable. Workable samples for different systems would have been appropriate, as would reproducing the 'Event Management' paper in the present volume.

getting picky:

The references are curiously split into two pages at either end of the volume. The first, 'Referenced Documents', is in neither the index nor the table of contents.

The Portability Guide seems to have been partitioned into as many volumes as possible, so you can pay as much as possible for them. This volume, Networking Services, at the very least should have been combined with Data Management (v5) and Window Management (v6). There are just too many small volumes to have littering one's desk.

conclusion:

On the whole, the 'man-page-per-function-with-crib-notes-on-the-protocol' approach used in the X/Open Portability Guide was easily followed. To draw a contrast, it is much quicker to understand, and more immediately useful, than the voluminous high level descriptive documentation supplied in the (already implemented) OSI 'isode' package. However, you could probably ignore the book given familiarity with ISO's transport specs, (IS 8072, 8073), or access to the 'isode' suite of libraries, the latter saving you the trouble of writing the code.

AUUG Book Club

Book Reviews

X/OPEN SECURITY GUIDE

by The X/Open Company Ltd
ISBN 0-13-972142-8, RRP $52.50, 1989

Reviewed by David Purdue
Kodak (Australasia) Pty Ltd
<auugn@munnari.oz.au>

As our machines become more widely connected, system security is becoming ever more important. It is with this in mind that the X/Open company has published its Security Guide, a supplement to the X/Open Portability Guide. The Guide discusses security and related issues for systems compliant with the X/Open Common Applications Environment, but almost all of it applies to any UNIX system, and many of the principles apply to any multi-user system.

The book starts by introducing some security concepts and terminology. The approach taken is that a computer system, including its hardware, software and the information stored on it, is a valuable resource, and the value of this resource must be protected. This value is reflected in the system's availability, the integrity of data and programs, and the maintenance of confidentiality.

Chapter 1 goes on to introduce the roles of those responsible for a system's security - the system administrator, the programmers and the users - and to talk about the various forms of security - physical, personnel, software, etc.

Chapter 2 discusses the mechanics of security, being those elements of an X/Open compliant system that affect the maintenance of security. These include users ID's, groups, processes, permissions and access rules, and privileges.

Chapter 3 talks about users of the systems, and their role in maintaining security. It covers the obvious things (e.g. don't leave a logged in terminal unattended), and also contains good advice on the importance of passwords and how to choose an effective one. Also mentioned are the ways users can use UNIX protection facilities to protect their own programs and data. This chapter contains many practical examples of the security implications inherent in using various tools, such as text editors, electronic mail and remote login.

Chapter 4 covers the responsibilities of programmers to not create programs that will breach security. Topics covered include programmer management, guidelines for writing secure programs, and the implications of writing programs that operate in a multitasking environment. Also covered is the writing of privileged programs, and special cases such as shell scripts and daemons.

Chapter 5 is on managing security, and covers establishing a secure installation, the ongoing management decisions involved in maintaining security - such as planning, choice of personnel and education - and how to deal with security breaches.

Chapter 6 gives more detail on system administration procedures for secure systems. This includes a programme for making the transition to a secure system, for administrating users within a secure domain, the security of the physical machine, file system security and network security.

The book is mainly a practical guide to operating a secure site, and as such it is quite good, but suffers from the consequent patchy presentation of the theoretical material that backs up this practice. Perhaps a better approach would have been to have a longer introductory chapter on the theoretical aspects, or to have given a list of references to such information.

Also because of the practical orientation, some of the statements made in the book are not justified. For example, in section 2.8, "Reasonableness Tests", one of the tests is that no process can gain write access to a directory. It is not explained why a process having write access to a directory is unreasonable.

One final criticism is that the book is a bit out of date. There is no mention of **sendmail** or the Internet protocols, and the implications of using those. This is surprising considering that the guide was published in 1989.

Overall, though, I think this book provides a good overall guide for day-to-day operation of a site that is concerned about maintaining security.

AUUG BOOK CLUB & PRENTICE HALL AUSTRALIA

20% DISCOUNT TO AUUG MEMBERS

Please send me copy/copies of the following books —

X/OPEN PORTABILITY GUIDE

☐ **Vol. IXS1 Commands and Utilities**
RRP $52.50* ISBN:1368-5835-X Paper 1989

☐ **Vol. IIXS1 Systems Interface and Headers**
RRP $52.50* ISBN:1368-5843-0 Paper 1989

☐ **Vol. IIIXS1 Systems Supplementary Definitions**
RRP $52.50* ISBN:1368-5850-3 Paper 1989

☐ **Vol. IV Programming Languages**
RRP $52.50* ISBN:1368-5868-6 Paper 1989

☐ **Vol. V Data Management**
RRP $52.50* ISBN:1368-5876-7 Paper 1989

☐ **Vol. VI Window Management**
RRP $52.50* ISBN:1368-5884-8 Paper 1989

☐ **Vol. VII Networking Service**
RRP $52.50* ISBN:1368-5892-9 Paper 1989

☐ X/OPEN SECURITY GUIDE
RRP $52.50* ISBN:1397-2142-8 Paper 1989

☐ X/OPEN SOFTWARE DIRECTORY
RRP $52.50* ISBN:1397-2134-7 Paper 1989

☐ *Hekmatpour*/C++ A GUIDE FOR PROGRAMMERS
RRP $35.95* ISBN: 1310-9471-8 Paper 1990

***Deduct 20% from listed retail price.**

Name:______________________ Organisation:______________
Address:____
(Street address only)
______________________________ Telephone:______________

Enclosed cheque for $ ________ (Payable to 'Prentice Hall Australia')
Please charge my: ☐ Bankcard ☐ Visa ☐ MasterCard
Credit Card No: ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐

Expiry Date: __________ Signature: ____________________

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

OR Use our FAST PHONE SERVICE call Liz Guthrie.
Have your credit card ready (9am to 5pm)
SYDNEY (02) 939 1333

Prentice Hall Australia Pty Ltd
7 Grosvenor Place, Brookvale NSW 2100
Tel: (02) 939 1333 Fax: (02) 938 6826
A PARAMOUNT COMMUNICATIONS COMPANY

West Australian Summer'90 Technical Meeting

Darryl K. Ramm
Pyramid Technology Corporation
South Shore Centre
83 The Esplanade, South Perth, 6151.
darryl@pyrmania.oz.au

The West Australian Summer'90 meeting was very successful, with over 70 delegates attending. The meeting showed a diversified range of interests, with talks covering topics such as data encryption, performance analysis, RDBMS systems, system evaluation and purchasing, lightweight processes and session management. One of the most useful functions of user group meetings is to just bring together the user community to allow exchange of ideas and information. Judging by some of the conversations occurring over coffee (and stronger refreshment later in the day) this part of the meeting was very successful. The added advantage of having interstate visitors helps to keep the local community up to date with what is happening elsewhere.

The keynote speaker was Ken McDonell from Pyramid Technology. For the benefit of the many West Australians who could not attend the AUUG'89 winter meeting Ken presented a repeat of his talk "OLTP - What's Behindthe Smoke and Mirrors". Ken reviewed the TP1 and Debit-Credit family of benchmarks, highlighting several methods used by vendors to produce large, and possibly meaningless, numbers of transactions per second. He then discussed more realistic benchmarks than TP1 and Debit-Credit being developed by groups such as the Transaction Processing Performance Council (TPC) an organization formed by over 30 database software and hardware vendors. Several other organizations such as the System Performance Evaluation Cooperative (SPEC) and UniForum, previously */usr/group*, are also developing more realistic benchmarks for measuring machine performance.

Ken described some enhancements to UNIX and RDBMS systems to support OLTP. These enhancements include fast locking, asynchronous but guaranteed I/O, modified CPU scheduling algorithms etc. More details are described in [McDonell89].

Chris McDonald presented a talk describing the development of a lightweight process library that he and James Pinakis are working on at The University of Western Australia. This development is aimed at simulating parallel concurrency on uni-processor systems for research into distributed operating systems. Chris and James have particular interest in Linda, a distributed operating system developed at MIT.

The presentation gave an overview of lightweight processes, or threads, and discussed their advantages over heavyweight processes as found in the conventional UNIX concept of a process. These advantages include fast process creation, termination, context switching and interprocess communication, but at some cost in user level software complexity. The talk described some public domain threads libraries that are available and some practical aspects of implementing a threads library (like how easier SPARC assembler is if you have a manual!). Chris and James talk was one of the more technical at the meeting and showed that there is good operating systems research being done locally.

Steve Landers of Functional Software discussed a session management package for ASCII terminals. The system utilizes the multi-page screenbuffers found in most modern terminals to provide rapid switching between sessions. Steve discussed the design alternatives that where available, comparing BSD job control, System V streams and BSD and System V pseudo terminals. He explained the justification behind the final design using System V sxt pseudo terminals. Steve explained the difficulty of dealing with badly behaved UNIX applications that do things such as unnecessarily setting the tty into raw mode, upsetting the session management control mechanism. Steve discussed System V streams based terminal I/O in some detail, pointing out that this would become more significant with the release of System V.4.

Michael Selig of Functional Software presented a talk entitled "Choosing a RDBMS - Do Your Homework". This talk was a chance for Michael to highlight some issues to consider before purchasing an RDBMS. He described the major types of RDBMS architectures including multi-process, single-server and hybrid architectures, and briefly noted some advantages and weaknesses of each architecture. Michael discussed the advantages of using a RDBMS based on standards like SQL, however he warned of the dangers of being trapped into vendors non-standard tools, embedded languages and 4GL's. He also described some of the RDBMS/UNIX performance enhancements such as raw I/O,asynchronous I/O and batch log commits and discussed the integrity and recoverability with RDBMS's, with some warnings to not assume that all systems have guaranteed recoverability.

Greg Rose of Softway presented a talk on public key encryption systems. Greg described a system devised by Rivest, Shamir and Adelman (RSA) which is about to be adopted by the Internet. Greg described how in a public key system the sender encodes the message using the recipient's public key. After transmission the message can be decoded by the recipient using his private key, but not by anybody else. In this way anybody with a directory of public keys can generate an encrypted message that can only be decoded by the intended recipient.

Greg pointed out that RSA encoding is fairly expensive in processor time as it involves exponentiating a 160 digit base number (the message) with an 80 digit encryption key. RSA offers the advantage of being a public key system and has potentially greater robustness to the more conventional Data Encryption Standard (DES). As the RSA system is public key it can be used for transmitting a small encoded private key with a message. Once the private key is decoded using the RSA system it can be used to decrypt the bulk of the message using a less expensive algorithm (such as DES).

Greg described how a public key encryption system also allows sender verification using a series of hierarchical verification signatures. The signature verification works by allowing senders authentication to be vouched for by an encoded signature of a "justiceof the peace". The identity of the "justice of the peace" is in turn verified by an encoded message signature of a more senior "justiceof the peace" and so on until some central authentication authority is reached.

Tim McGrath from Port Community Systems and Ian Crawford from the Fremantle Port Authority presented an entertaining two person presentation on managing the evaluation process for the purchase of acomputer system. The talk had particular emphasis on government departments, but much of the suggestions and advice would apply to any potential customer. Tim and Ian covered initial budgeting, choosing consultants or planning committees, planning the request for tender (RFT), specifying requirements, benchmark design, details of publishing the RFT, making the tender evaluation, contract negotiation, systems installation, maintenance and support arrangements and much more. To borrow their metaphor, the talk led from the first introduction between the eager potential partners, through the courtship stages, wedding, honeymoon and finally living together. This talk raised some interesting points in an amusing way. Many people in the audience could sympathize with the difficulty of performing system evaluation and find useful some of the suggestions in their talk.

Ken McDonell concluded with a retrospective view on a decade of life with MUSBUS (Monash University Suite for Benchmarking UNIX Systems). From its early days at Monash and its acceptance within Australia, to its acceptance internationally by organizations such as EUUG and numerous UNIX system vendors. Ken described the MUSBUS architecture, especially how it controls the simulated multi-user workload and performs some important error checking. He pointed out the importance of specifying an appropriate workload so that the benchmark is simulating as closely as possible the desired environment. Ken also described some new features that he will be including in the next, and final, version of MUSBUS. These include a load simulator that allows a work profile to be specified in low level constructs such as CPU andI/O usage.

The organization of the West Australian Summer'90 Meeting was largely the work of Glenn Huxtable and Chris McDonald. Thanks go to them and the others who helped make the meeting a success. Judging by the interest shown at the West Australian Summer'90 meeting the Summer'91 meeting should be a guaranteed success.

[McDonell89] McDonell, Ken J., "OLTP Performance - What's Behind the Smoke and Mirrors", *AUUGN*, 10, 4 (1989).

Using Unix as a Persistent Programming Environment

A. J. Hurst
Monash University

Introduction

Persistent programming denotes a philosophy of programming in which all program objects have the same data rights. In particular, one important data right of a program object is its lifetime. Most languages control lifetimes of data objects by declaration and scoping rules, but one important exception is the data object *file*. Whereas all other objects normally are destroyed when a program terminates, a file may outlive the program that creates it. Programmers have exploited this mechanism in order to preserve important data, but it necessitates a tedious process of converting from input representations to internal representations, and then back again for output. It has been estimated that this activity accounts for some 30% of program code.

Persistent languages accord all objects the same lifetime rights, so that no conversion from external to internal form is required. This mechanism is akin to virtual memory, where the transfer of objects between internal and external stores is managed automatically. However, persistent systems usually give the programmer more explicit control of such movement, without requiring explicit conversions. Data objects can then be said to be *persistent*, meaning that they have the right to outlive their parent environment, and this right is conferred independently of the data object's other attributes.

In order to implement persistent systems, a form of virtual memory is required. This may be done by means of capability based systems, such as has been developed at Monash University. If persistent systems are to succeed however, implementations on conventional architectures are also required. The paper develops the important point of how the underlying persistent mechanism reduces to the problem of mapping names to objects, describes the processes involved in implementing persistence, and shows how the Unix file system can be used as a tool for achieving this. In addition, some experiments with using a graphical interface to both a persistent system, and to the Unix file system, are described.

Persistence

Persistence is a term used to describe the lifetime of program objects. Just as such objects have other attributes such as type and locality, over which programmers have control, so too should the lifetime of an object be under the control of the programmer. Lifetimes range from the very short, such as local variables in an innermost procedure call, to the intermediate, such as global variables in a program, to the long and very long, such as data and system files. Indeed, in most general purpose languages, files are the only objects with arbitrary lifetimes, able to persist beyond the lifetime of the creating environment.

The persistent programming paradigm provides all program objects with this ability to outlive the program that creates them. Without this ability, the programmer is forced to convert from internal data types, such as integers, lists, arrays, etc., to external representations, usually just character strings. Of particular concern is the fact that pointers cannot easily be mapped to external forms, and subsequently reconstructed when the object is read back in. This necessitates mapping the internal structure of a list, tree, etc., into some flattened form as a character string that can be stored in a file. When the object is restored in a later program environment, this conversion must be inverted. The programming effort required for these processes is not trivial, and is very error prone.

When the programmer defines the various programming objects of a program, it is important that this can be done without regard to the persistence attribute of the object. For example, declaring local objects should be done with respect to the attributes of the object in the same way as for a global object with the same attributes. If changing an integer variable from local to global required a change in the type declaration, the language would become unecessarily cumbersome.

In the same way, persistent languages allow program objects to be declared and created in much the same way as for conventional languages. Indeed, there is no requirement for a programmer to actually use persistent features.

Persistence can be thought of programmer controlled virtual memory. Objects move between the primary store, or program space, and the secondary store, or persistent space, without change of representation. The process of moving data between the two spaces is transparent, like virtual memory, and the objects themselves are independent of the store boundary, and can be copied back and forth just like segments or pages in a virtual memory. Having to recast objects as new structures is as obsolete as linking in new code overlays in non-virtual memory systems.

Persistence is not a new idea. Some programming languages (or more properly, programming language environments) have provided for persistent storage of objects. APL and LISP are notable examples, providing in many implementations the notion of a workspace, which can be dumped to secondary storage and subsequently restored. These workspaces can contain variables, function or procedure declarations, and other program objects. The major disadvantage with these early forms of persistence is the fact that the programmer has little control over which objects are persistent: either nothing is (no workspace is saved), or everything is (the workspace is saved). What is required is the ability to specify the lifetimes of objects individually. In more formal terms, we demand that *persistence is an orthogonal attribute of program objects.*

The concept of persistence as an independent attribute of program objects has been touched upon by several researchers, but it was Atkinson [1] who first identified the term.

Programming with Persistence

Just as with other programming models, persistence requires a certain discipline on the part of the programmer. Forcing all objects to be persistent, as happens with the APL/LISP models, is too heavy handed an approach. We need mechanisms that will allow the programmer to control which objects are persistent, and how they are used within programs. To see why, we must first examine the basic persistent programming model.

Objects that are to be persistent are simply objects that may be retrieved in later, different executions of either the same, or different, programs. To make an object persistent we might simply add a **persistent** keyword to its declaration. Programs creating new persistent objects might declare *x* : **new persistent array** *[1 .. 6]* **of integer**, while programs reusing old persistent objects might declare them as **persistent procedure** *y*.

Doing this requires some form of nomenclature that is agreed upon by the two program executions. One obvious way is to use the identifier attached to the object. For example, if *x* is an integer variable made persistent in program A, then when program B declares a persistent variable with identifier *x*, the connection is established, and the same object (with the same type) as was used in program A is available for use in program B.

However, this simple scheme suffers from obvious disadvantages. How does the compiler know when compiling B that the type of *x* is integer? What if there already is an identifer *x* in program B? What would happen in there already was a (persistent) *x* when A declared its use of *x*? How could B access two different persistent objects both named *x*?

The answer to these problems lies in giving different persistent names to persistent objects, and binding these names to local identifiers within a program. This is nothing more than that which already happens with those well known persistent objects, files. When a file is opened, a binding is established between the file name, a system wide, persistent name, and the file variable, identified by a name local to the program.

Environments and Context Sensitive Addressing

Just as in conventional file systems, a flat naming structure is inappropriate to persistent stores. The consequent name explosion generated by having to find new, unique names for each persistent object makes flat, or global, naming cumbersome. Just as hierarchical naming has taken over in the file domain, so most persistent systems offer a mechanism to structure the persistent store so that names are *context sensitive*, meaning that they are interpreted according some context in which they are found.

Take Unix as an example. The solitary file name *x* will interpreted as a context sensitive name, relative to the current working directory. It will not be confused with any other file name *x* defined in any other directory. The directory structure is a tree, which is rooted in the file name /, also known, appropriately enough, as *root*.

Since all files are either leaf nodes or directories, the file system forms a hierarchical tree. But note that Unix goes further than this, as files may be reachable from more than one directory. The Unix command `ln` allows the programmer to establish a new name for a file, in a possibly different directory. The same file may thus be accessed by different paths starting from root, and the file system is no longer a tree, but a directed graph.

This flexible and convenient form of naming allows programmers to establish localities of related objects, so that files related to a common task are accessible directly within a single 'working directory'. Software can be structured so that it need not know absolute file names, and thus becomes more flexible in operation.

Such addressing we call *context sensitive* addressing [2], as it depends upon the current context (or current working directory, in Unix terms) when the name is translated. The context or 'locality' gives the user (and computer) a very much restricted domain of objects, and prevents the user (if not the computer) from becoming overwhelmed by too much data. As Dijkstra has pointed out, we are limited by 'our inability to do much' ([3], p.1).

Such locality is an important tool in much of computing. For very similar reasons, it is convenient to organize the persistent store in the same way. However, we have traded one form of naming convenience for another. Because the file name no longer represents a unique identifier for the file, we must introduce another, necessarily unique, name. The reason that we need such a unique name is that the file has an existence independent of the naming path used to reach it. Such a name we shall call the *persistent identifier.*

Capabilities as Persistent Identifiers

The important attribute of a persistent identifier is that it is a once-and-for-all-time label attached to a persistent object. When the object is created, a new unique persistent identifier is also created. When the object is finally destroyed, or becomes inaccessible, the persistent identifier is discarded.

To some extent, the persistent identifier is like a pointer. It serves as a unique handle to reach the object, and cannot (or at least, should not) be used to reach any other object. When the object is created, a memory allocation routine returns a pointer value to denote the storage space allocated to that object. This storage space is independent of any other storage space for any other object, although often a knowledge of the allocation strategy allows unscrupulous programmers to get at other objects via such pointers.

For some time it has been recognised that pointers are potential sources of trouble in programmed systems. Various attempts to control them have been made, but the most successful has been the notion of regarding pointers as *capabilities* [4]. Important to the notion of pointers as capabilities is the fact that pointers usually undergo some form of address translation on modern architectures, and that this translation can be exploited to ensure that only legitimate pointers translate successfully. Various schemes exist to do this (see, for example, [5]), but basically, they all reduce to making it difficult or impossible to forge a capability. In the Monash Multiprocessor ([6]), a randomly generated *password* is added to so increase the redundancy of a capability, that it becomes very unlikely that anyone could guess a legal capability.

Capabilities make excellent persistent identifiers. They are so closely coupled with the mechanism for accessing an object, that they can almost be considered to be the object. The process of retrieving a persistent object then becomes one of uttering a capability, and asking the capability translation mechanism to make the object accessible. Access rights to the object can be encoded in the capability, so that objects can be read-only, or execute-only, and even made type secure, meaning that only certain well defined operations or procedures may be applied to the object when it is accessed ([7]).

Unix File Names as Persistent Identifiers

Unfortunately, not all architectures are capability architectures. Any language of system designed around special hardware will suffer a severe acceptance problem if it is not also available on conventional machines. Persistence is claimed to be an important general purpose programming tool. It is necessary therefore to investigate how persistent mechanism can be implemented on conventional architectures.

One of the first persistent languages, PS-algol ([8], [9]) used a soft implementation of persistent identifiers, represented as a 32 bit integer, and performed software translation of this persistent identifier, or *pid*, on the fly. To improve performance, the pid was translated only the first time it was encountered, and was then replaced by the primary store address of the referenced object, which was also brought in from persistent store if necessary.

One special class of object, called a *database*, contained mappings from identifier strings to persistent identifiers, allowing programs to browse through the persistent store by referencing such database objects. Since these were standard types in the language, arbitrary graphs of persistent objects could be constructed.

The behaviour of these database objects can be modelled by Unix files. The Unix file system provides a name translation service, for mapping identifier strings to file objects. Since files can be directories, this structure can be used to represent exactly the same form of persistent store provided by PS-algol. Of course, Unix misoneists may well argue that the persistent structures just model the Unix file system!

Persistent identifiers *per se* are no problem. These can be generated by time stamping a stub file name. Each time a new persistent object is created, and written to a file, the file is linked into a master directory of persistent objects with the unique file name, such as `pid-90-01-12:15:21:56`. Local names can map to this file by linking to it, or by storing this file name as a character string within the local file. Assembly language programmers will recognise this as a high level form of indirection. Since the Unix persistent identifier is just a string of characters, it can be used in much the same way as a capability, and extra characters used to encode rights information, etc.. Of course, the read/write protection bits of Unix could themselves be used for that purpose. The only disadvantage of this scheme is that because the name lookup is performed in software, its performance is comparable to the PS-algol scheme, but it cannot compete against the hardware translation mechanisms of the Monash Multiprocessor.

A Graphical Browser for Persistent Objects

To illustrate just how closely the persistent programming model can be supported by Unix, a serendipitous result from a recent student project can be used. A graphical browser for persistent systems was developed by Wong [10]. This had as its aims the construction of a graphically based interactive tool, that could be used to explore the structure and value of objects in a persistent store. One of the problems with persistent stores (and this includes conventional file systems) is the difficulty of keeping track of what is stored where. As we have already seen, using a hierarchical system somewhat ameliorates this problem, but the real insidiousness is demonstrated by programs such as `which` and `find`. How many Unix users have forgotten where in their directory structure resides an old, not-recently-used file that they now wish to use?

The graphical browser (itself developed from an earlier, text-based, browser) allows the user to point-and-click with a WIMP (Windows, Icons, Menus, Pointer) style interface, running on a Sun under Unix 4.3 and Suntools. Upon startup, the root environment is shown. By pointing at, and clicking on, the icon representing the root, the objects referenced by entries within the root are themselves displayed iconically. The user is able to rapidly traverse the persistent store, by successively tracing down paths, backtracking, or just selecting new branches not previously visited. The window manager takes care of the display, by scrolling the window upon which the icons are displayed, so that the current focus of attention is always visible. Of course, the user can also scroll the window explicitly. The icons are defined by the type of the persistent object represented, so that a strong visual clue is given as to what the object is, thus often freeing the user from having to interact further to determine whether the displayed object is the object being pursued.

The browser does not talk directly to the persistent store, as it was designed to handle a variety of persistent store sub-systems. Instead, it relies upon a *persistent store server* to carry out the basic operations of fetching persistent store objects. Because the browser is just a browser, there is no need to write to the persistent store, but if an editing facility were to be included as a sub-component of the browsing system, this would obviously change.

The serediptious result referred to earlier was the observation, after the browser had been completed, that, just by changing the interface between the browser and server, the browser system could be used as a front end to a Unix file store. The icons now represent files, with the different attributes of a file reflecting themselves in different icons. For example, a directory is displayed as an iconic tree with a table at a leaf node, while a file is displayed according to the file type returned by the `file` command in Unix.

Such use of the browser is possible because it follows an important principle of software engineering: information hiding. The browser does not need to know exactly what happens when it issues requests to the server, and therefore this information should be hidden from it. Because the two systems are coupled only through the interface, the server implementation can change significantly, from an arbitrarily typed-object persistent store to a conventional character-stream file system, without any redesign of the browser.

Conclusions

We argue that persistent programming will become a significant programming paradigm. Like many advances in programming, the paradigm draws from conventional experience, and extends our understanding of the programming process by simplifing and generalizing a key concept. That concept is that program objects can have lifetimes that are independent of the other attributes of the object.

In addition, the work has demonstrated how the reuse of existing programming facilites can be used to advantage. Always an important part of the Unix philosophy, software tool construction by building upon existing components allows new facilities to be created with a minimum of effort. The rapid prototyping of a graphical Unix file browser was possible because of adherence to software engineering principles, namely information hiding and software reuse.

Bibliography

1. M. P. Atkinson, K. J. Chisholm, W. P. Cockshott; *Nepal - the New Edinburgh Persistent Algorithmic Language*, in Database, Pergammon Infotech State of the Art Report, Series 9, No. 8 (January 1982).
2. A. J. Hurst, *A Context Sensitive Addressing Model*, Proceedings of the 10th Australian Computer Science Conference, Deakin, Feb 1987.
3. O.-J. Dahl, E. W. Dijkstra, C. A. R. Hoare; *Structured Programming*, Academic Press (1972).
4. R. S. Fabry; *Capability Based Addressing*, Comm. ACM, vol. 17, no. 7, pp.403-412 (July 1974).
5. M. S. Anderson, C. S. Wallace; *Some Comments on the Implementation of Capability Systems*, Australian Comp. J., vol. 20, no. 3 (1988).
6. M. S. Anderson, R. D. Pose, C. S. Wallace; *A Password Capability System*, Computer J., vol. 29, no. 1, pp. 1-8 (1986).
7. W. A. Wulf, R. Levin, S. Harbison, *Hydra/C.mmp: An Experimental Computer System*, McGraw-Hill, New York (1980).
8. M. P. Atkinson, P. J. Bailey, K. J. Chisholm, W. P. Cockshott, R. Morrison; *PS-algol: a Language for Persistent Programming*, 10th Australian Computer Society Conference, Melbourne (1983).
9. W. P. Cockshott, M. P. Atkinson, K. J. Chisholm, P. J. Bailey, R. Morrison; *POMS: A Persistent Object Management System*, Softw. Pract. and Exper., vol. 14, no. 1 (January 1984).
10. G. Wong, *Persistent Editing Environments*, Honours Project Report, Department of Computer Science, Monash University, 1989.

UNIX International

UNIX System V Release 4 and OSF/1

A Comparison

UNIX System V Release 4 and OSF/1

A Comparison

Executive Summary

UNIX System V Release 4.0 (SVR4) is available for general release and is being tested on over thirty different vendors' hardware platforms. UNIX International, in conjunction with AT&T's UNIX Software Operation (USO), has been developing market requirements for future System V releases. These will be made public via the distribution of the UNIX International *Product Roadmap*, the foundation product planning document for UNIX System V.

This paper has been provided to help the industry determine the relative merits of UNIX System V versus a future product from the Open Software Foundation™ (OSF™) called OSF/1™. There has been much speculation in the press about OSF/1, but little is really known. Critical issues, such as the kernel architecture, remain unresolved. Real OSF/1 knowledge will likely be scarce until it is introduced, reportedly in the last half of 1990. This paper briefly discusses advances in SVR4, makes comparisons with OSF/1 where information is available, and discusses important issues which should be considered as more is known about OSF/1.

SVR4 was designed to unify and merge the several major derivatives of the UNIX system: UNIX System V Release 3, the Berkeley Software Distribution (BSD) and Xenix™. With this unification, UNIX System V Release 4.0 is compatible with over 80% of the installed base of UNIX system software, over twelve thousand applications. In addition, SVR4 contains substantially enhanced functionality as compared to UNIX System V Release 3 and other flavors of the UNIX system. However, it is even more important to note the advantages of SVR4 in its overall engineering. The infrastructure for the future evolution of UNIX System V is built into SVR4. The modularity of the various functions in SVR4 ensure that continued improvements in one area will not have a negative effect on another area. Furthermore, SVR4 is based primarily on time-tested code that has run on a wide variety of hardware architectures, and has had its bugs systematically eliminated.

The OSF equivalent, OSF/1, whether based substantially on IBM's AIX™ product, other existing software (e.g. Mach), or new code, must first 'catch-up' with SVR4, and do so with code that has yet to stand the test of time that comes after having been operating on dozens of different vendor systems over many years. Only then will OSF/1 be able to address future needs, and then only if its users do not demand 'catching-up' with the newer System V releases.

The greatest strength of UNIX System V is not that it provides every feature known, but that it provides a strong environment from which a user can build the right applications or add new technology. This allows the market to develop and choose among competing solutions that exist on top of System V rather than having to accept a unilateral choice by a controlling developer, even if that developer has requested technology about different approaches.

This paper compares SVR4 and OSF/1 by examining what is known, via press reports, about OSF/1 and its 'base' code, AIX. The comparison first discusses general open systems requirements and then basic operating system functionality.

UNIX System V Release 4 is real and being ported to dozens of vendors' systems today. It maintains compatibility with over twelve thousand existing applications and provides a solid foundation from which to address future market requirements. SVR4 advances System V's open systems leadership by creating a vendor-neutral software platform that end-users can use to mix and match software and hardware within a virtually seamless operating environment. We believe that no other system can or will offer such complete capability.

1. Introduction

This document attempts to compare OSF/1 and UNIX System V Release 4.0 (SVR4.0). This is difficult because the first version of OSF/1 has not been finalized. OSF statements suggest OSF/1 will be based substantially on IBM's Advanced Interactive Executive (AIX) operating system therefore the following comparison is often a comparison of SVR4.0 and AIX. While it is necessary to use AIX as the basis for comparison it must be recognized that since OSF will be tailoring AIX to its needs, the result may be significantly different from the current AIX product. Additionally, OSF/1 will most likely be based on AIX version 3.0, a product which is also yet to be delivered to the marketplace.

The following discussion first compares SVR4 and OSF/1-AIX on the general basis of open systems requirements. Next, basic operating system functionality is discussed followed by brief discussions on multiprocessing, security, and networking.

2. High Level 'Open Systems' Comparison

Both OSF/1 and SVR4.0 claim to be Open Systems. This comparison looks at how well they support that claim. Both UNIX International, and the Open Software Foundation use the characteristics of compatibility, portability, scalability, and interoperability to define an open system. This section uses these criteria to compare SVR4.0 and OSF/1.

2.1 Compatibility

Compatibility provides a source-code level commonality which allows applications to be run on any UNIX System V system without changes besides recompilation. Compatibility means an application developer can move their software investments to a different hardware platform without any code changes.

2.1.1 Source-level Standards

In order to provide compatibility, an open system must support a consistent set of standardized application programming interfaces. For the application programmer the most significant interface is the primary system interface which in the UNIX world has been historically known as the SVID (System V Interface Definition). Because of the importance of this definition, IEEE has created a standard, influenced significantly by the SVID, referred to as 'POSIX'. This standard has received significant support via government purchasing standards (the NIST FIPS) as well as from X/Open. X/Open is also standardizing in areas beyond POSIX to create their Common Applications Environment (CAE) via the *X/Open Portability Guide (XPG).*

Both SVR4.0 and OSF/1 will support the POSIX and XPG3 industry standards. Other than availability dates, (i.e. SVR4 is compatible with POSIX and XPG3 today), there is unlikely to be much difference. But what about the installed base of software existing before these standards?

2.1.2 Preserving Past Software Investments

Support for 'de-facto' standards, specifically Microsoft's XENIX™ system, UNIX System V and Berkeley BSD 4.2/4.3, will most likely show more variation. Compatibility with these 'de-facto' standards was a major reason for the development of SVR4. SVR4 was designed to unify and merge these major derivatives of the UNIX system so that SVR4 would be compatible with over 80% of available UNIX software, some twelve to fifteen thousand applications. SVR4 provides complete source-level compatibility for Microsoft's XENIX system - plus binary compatibility - and a high degree of application source-level compatibility for 4.2 and 4.3 BSD, and, of course, it will maintain complete compatibility with the standard UNIX System V programming interfaces that have been supported in the past.

OSF intentions in this area are unclear. AIX 3.0 is SVID release 2 compatible and has been described as BSD compatible, but there has been no clear indication of compatibility with the installed base of Xenix software. And what about SVID compatibility with SVR4 or future SVIDs? Unknown, is to what degree OSF/1 will maintain compatibility with UNIX System V features that are not in XPG3 or POSIX. Surely the fact that System V defines past compatibility and is continuing to make advances, would suggest that System V will always play a major role in defining POSIX and X/Open standards.

2.1.3 User Interface Compatibility

Another critical aspect related to compatibility is the application programmer interface (API) for the Graphical User Interface (GUI). UNIX System V Release 4 supports the X Window System™ and thus can support any GUI based on X, which includes OPEN LOOK and OSF/Motif.

2.1.4 System Software Compatibility

SVR4.0 extends compatibility by defining interfaces for device drivers, STREAMS I/O modules, and Virtual File System (VFS) types. These open up the possibility for vendors to develop compatible software in areas - like device drivers and communication software - where compatibility has never before been possible. OSF/1 will undoubtedly have an analog to VFS but it is not known whether OSF would choose to use STREAMS I/O or some equivalent capability. If OSF/1 does not support STREAMS it will be incompatible with the numerous applications and development tools that are already taking advantage of this powerful feature.

2.2 Portability

Portability is similar to compatibility but refers primarily to allowing application binaries to operate on a variety of vendor systems. This would give a capability similar to the 'shrink-wrapped' software found in the PC market and thus customers could purchase application software and use it 'as-is' on computer systems from a variety of different vendors. UNIX System V is addressing this need by developing Application Binary Interfaces (ABIs).

2.2.1 Application Binary Interfaces

AT&T has been working with manufacturers to establish Application Binary Interfaces, which define a set of interfaces that allow compiled software to be ported in binary form across different vendors' systems that share a processor architecture. An SVR3.2 ABI already exists and is successful for the Intel 386 architecture. SVR4.0 ABIs for many processors are already under industry review, and ABIs for Intel 80386, Motorola 68000 family and 88000, WE32000, and SPARC processors will be generally available by the end of 1989.

OSF has put out a Request for Technology (RFT) for an Architecture-Neutral Distribution Format (ANDF), which would provide a "single format for distribution that is hardware-independent." This technology would define a single intermediate format usable with many architectures. But to be useful, ANDF technology must solve issues concerning performance, ease-of-support, and ISV source code protection. For more information, see the UNIX International white paper "Mass Distribution of UNIX Software".

It is important to note that ANDF requires the ANDF software to be translated into a binary application that is specific to the target architecture of the system for which the software is being installed. The result of that translation step could itself conform to the applicable ABI for the target architecture. Therefore, the future existence of an ANDF does not in any way diminish the need for ABIs now or in the future.

OSF's ANDF RFT stipulates that candidate technologies must be available for shipment during 1990. The ANDF technology is still in its infancy and it does not seem likely that it could become part of OSF/1. Since OSF has not announced compliance with the AT&T initiated ABIs, they may or may not choose to make it possible for vendors who license OSF/1 to also support the ABIs. Compliance, by OSF, with the AT&T ABIs would make life much easier for software vendors.

2.3.2 XENIX System and MS-DOS Compatibility:

On the Intel 80386 architecture, SVR4.0 provides binary compatibility with XENIX System applications. The degree of XENIX binary compatibility provided by OSF/1 is unknown at this time. Several vendors offer software that provides MS-DOS compatibility under the UNIX System. There is no reason to believe that these same vendors will not provide this capability for OSF/1 systems.

2.3 Scalability

An open system must be hardware independent and available on a wide range of hardware platforms from a variety of hardware vendors. Scalability means that customers can keep the same software environment while upgrading their hardware platform or changing vendors.

UNIX System V runs on a wider range of hardware platforms (from PC's to super-computers) than any other operating system, and is supported by every major hardware vendor, most of them choosing UNIX System V. Scalability is further enabled by the existence of System V ABIs.

Obviously, OSF/1 has yet to run on any system large or small. Even AIX has limited scalability experience. While IBM has versions of AIX running on 370s, PS/2s and RTs, they are not compatible. There has been considerable experience porting System V to hundreds of different computer systems while it is unclear how easy it will be to port an AIX-based product. Lack of any ABIs will also affect the scalability of OSF/1.

2.4 Interoperability

An open system must support communication and interconnection among a wide range of applications. Interoperability addresses the question of:

- how users interact with applications,
- how applications work with each other, and
- how one system communicates with another.

Interoperability allows customers to pick hardware and software from multiple vendors and have the pieces work together.

SVR4.0 supports a wide range of networking standards under a single standard interface (Transport Interface, also known as TLI). It also provides Remote File Sharing (RFS) and Sun's Network File System (NFS), which allow application transparent file sharing among different systems, including systems that are not based on the UNIX System. NFS has already been widely adopted by computer vendors and end-users alike. The NFS Remote Procedure Call (RPC) capability is already being used by ISVs to create distributed applications, even between System V and MS-DOS systems. The Transport Interface allows applications to be network and protocol independent and supports easy migration to the OSI protocols.

OSF/1 will support TCP/IP using the BSD sockets interface, which SVR4.0 will also support, but it is not as portable an interface as STREAMS. OSF does plan to support X/Open's version of TLI, called XTI, as will System V. Beyond that, it is not clear whether OSF will support NFS or IBM's Distributed Services (DS) remote file system.

3. Base Operating Functionality

3.1 Kernel memory management and data structures

The management of kernel memory and data structures is crucial to the ability of the system to scale from PC's to supercomputers. System V continues to be the only software that spans so wide a range. SVR4 contains many improvements to the kernel to support both smaller and larger platforms.

Most kernel data structures grow dynamically by increments, rather than being preallocated in real or kernel virtual memory. In SVR4, the same binary system code can run divergent workloads and immediately adapt its data structures to changing system demands. This policy imposes the least demands on low-end systems, yet allows expansion to high-end systems with the same software. The AIX approach to memory management is less flexible, and requires larger kernel page tables than SVR4. To attempt to get system scalability, AIX preallocates the maximum sizes of system data structures in kernel virtual memory, then uses kernel paging to save memory.

The SVR4 kernel uses the virtual memory hardware to good advantage in providing a common memory pool for program text and data as well as file information. It does so without requiring special hardware support or large kernel virtual address spaces, which makes very effective use of physical memory. The flexibility of this approach is that both the kernel and the application can dynamically map files into their address spaces and use only as much space as is needed. AIX is likely optimized for the virtual memory hardware of the 386, PC/RT, and 370/XA. It is not clear how portable it will be to other architectures. Machines with limited kernel virtual address space, such as the VAX or 68000, may suffer substantial performance penalties.

SVR4 has improved the organization of system data structures to support large numbers of users and larger applications and services. For example, linear searches of data structures have been eliminated. Applications can now have an arbitrary number of simultaneous open files. AIX claims similar improvements, but has no advantage in this area.

3.2 Virtual Memory

SVR4.0 uses a virtual memory architecture based on the SunOS Operating System. It is considered the best virtual memory architecture available today. It implements a "single-level store," which lets processes access files and devices as ranges of bytes within the virtual address space of the process. Programs can now access and share data by mapping files or parts of files into their address spaces. The common memory pool provides a consistent view of data, whether accessed by file or memory operations. Maximum flexibility is provided by permitting page-level operations. There are no arbitrary limitations on the portion of the file to be mapped or on the number of files that can be mapped. The AIX VM is older technology, and deals only with segments and whole files. AIX limits you to 10 segments of 256 megabytes each. Consistency is provided only when it is used with the AIX file system.

In SVR4, swap space is more flexible and scalable, because paging is done to files instead of to preallocated swap areas. The swap areas are managed to provide proper resource control by notifying programs of space constraints when memory is allocated instead of arbitrarily terminating a process when space is exhausted. AIX uses preallocated swap areas and "overbooks", which can cause arbitrary process termination when the system runs out of memory.

The SVR4 virtual memory mechanism allows libraries and other code to be shared among programs. The ability to map libraries at arbitrary addresses allows programs complete control of their address spaces. Shared libraries are marked with version numbers that allow old binaries to run on new systems ensuring maximum compatibility between different releases. It is not clear how AIX or OSF/1 will address this issue.

OSF/1 may replace the AIX virtual memory manager with the Mach memory management implementation developed at Carnegie-Mellon University, which is an improvement over the AIX implementation, because it is more portable and flexible.

3.3 File System

SVR4.0 offers the vnodes architecture, which supports multiple file system types. SVR4.0 will ship with seven different file system types, including UNIX System V, UFS - which is based on the BSD Fast File System, and '/proc', which offers a unique interface to running processes. Supporting multiple file systems allows system vendors and users to choose the best file system for their application environment. Since different file systems make different tradeoffs, like performance versus flexibility, having a choice can be very important. Such a capability also allows the development of application specific file systems, for example, a UNIX file system which can read and write MS-DOS diskettes.

OSF says that OSF/1 will have a file switch and will support UNIX System V and BSD file systems, plus a file system from AIX that provides database journaling. It is not clear whether AIX will support the BSD file system as well as it supports the AIX file system.

3.4 Real-Time Support:

The SVR4 kernel has been substantially improved to increase its responsiveness for real-time applications. Real-time support makes System V a better platform for time or event critical tasks found in applications such as transaction processing or factory automation, where the system must be determinably responsive to high priority needs.

SVR4.0 provides many more kernel preemption points than previous UNIX System V releases to accomplish the same task. The insertion of many preemption points permits lower interrupt latency without introducing the overhead of locking for non-real-time environments. OSF/1 is stated to have a preemptive kernel to increase response time predictability. The AIX preemptive kernel is an attempt to reduce interrupt latency. But because AIX requires locks to ensure system integrity, an interrupt can still be delayed by kernel code that holds a lock.

SVR4 supports high-resolution timing to the granularity of nanoseconds. It also provides a new real-time scheduler with dynamically-settable parameters, to allow high-priority processes to preempt lower priority or interactive programs. OSF/1 will also have a real-time scheduler. Generally it appears that both operating systems will need additional work to become full-fledged "real-time" operating systems.

3.5 Internationalization

Both SVR4.0 and OSF/1 will support the internationalization capabilities specified in ANSI X3J11 C and the *X?Open Portability Guide* Issue 3. Beyond that, SVR4.0 provides some, but not all, of the capabilities of the Multi-National Language Supplement (MNLS) Release 3.2. System V already supports a number of European languages such as German, French, Italian etc. In addition, the UNIX International Internationalization work group is developing requirements for supporting Asian languages. Please refer to the UNIX International paper "Work Group Update" for more details.

4. Multiprocessor Support

OSF is currently evaluating whether to support the Mach multiprocessing kernel. Combining Mach with AIX, while not impossible, is a difficult task at best and will clearly increase the amount of time required to stabilize the first OSF/1 release. Unfortunately, the result is adding an advanced capability to old functionality.

Future System V multiprocessing, as defined by the UNIX International Multiprocessing work group, will be implemented on top of the advanced foundation and functionality provided by SVR4. The work group, which contains over 30 members, including representatives from Carnegie-Mellon's Mach group, is near finalization of a System V multiprocessing specification. This specification includes the capability to provide symmetric multiprocessing with significant Mach-like constructs. Please refer to the UNIX International paper "Work Group Update" for more details. It is also important to note that UNIX System V has a long history of multi-processor ports, and AIX has no such history.

5. Security

Security requirements, while formalized by the government market, are critical for the commercial market as well. Today, AT&T's Multi-Level Security (MLS) product, equivalent to a B1 rating, is available on UNIX System V Release 3.2. UNIX International and USO are committed to advancing security functionality, for both commercial and government markets, in all future System V releases. UNIX International and USO are currently finalizing a future System V release which will support a B2 or higher security rating. It is not clear how soon OSF/1 will provide a B2 level.

6. Networking Support

SVR.4 continues to use and enhance the STREAMS I/O system introduced in UNIX System V Release 3. The Transport Interface, discussed earlier, provides protocol and network independent services at the Transport Layer of the OSI reference model. Having such modular interfaces allows System V to transparently support all major networking systems such as:

1) The worldwide UNIX TCP/IP network.
2) Proprietary networks such as IBM SNA.
3) OSI networks.

Applications can be developed without concern for the underlying network and will be compatible with whatever network a user chooses. This allows a smooth transition path from existing networks to the ISO protocol suite. OSF/1 will provide TCP/IP on the BSD sockets interface, but STREAMS support has not been announced. Without STREAMS, it is not clear how easy it will be for OSF/1 to support existing applications and provide a smooth transition to OSI.

7. Conclusion

UNIX System V Release 4 is real and being ported to dozens of vendors' systems today. It maintains compatibility with over twelve thousand existing applications and provides a solid foundation from which to address future market requirements. SVR4 advances System V's open systems leadership by creating a vendor-neutral software platform that end-users can use to mix and match software and hardware within a virtually seamless operating environment. We believe that no other system can or will offer such complete capability.

Optimizing the B5FS File System

Steven Bodnar
*steve@labtam.oz.au**

Stephen Prince†
sp@labtam.oz.au

Labtam Information Systems Pty Ltd
Braeside, Victoria 3195

ABSTRACT

The B5FS file system is a reimplementation of the Berkeley Fast file system technology within the framework of the System V file system switch, whilst retaining System V.3 file system semantics [Prince88a].

The following paper describes some macroinefficiencies which have been isolated in the B5FS implementation, that cause cache and overall performance to be less than its System V counterpart, which is refered to in this paper as the S5 file system. That is, we describe the issues which needed to be addressed, in order for users not to feel penalized when working with small files on the B5FS file system.

1. Background

Following the initial development of the B5FS file system, it was discovered that a hybrid system containing both B5FS and System V file systems showed that B5FS while exhibiting the same performance characteristics as a Berkeley file system was noticiably slower when file sizes were small enough to be contained in the system's buffer cache.

This paper discribes the issues which needed to be addressed, in order for users not to feel penalized when working with small files on the B5FS.

2. Methodology

All tests were performed on newly made file systems. This paper does not show the System V file system inadequacies in handling a greatly used file system.

The effect on performance as a result of changes to B5FS were measured using the *fstime* portion of the MUSBUS 5.2 benchmarking suite [McDone87a] with zero interactive users‡. *fstime* measures the rate at which user programs can transfer data to or from a file without performing any processing on it. The measurements were run three times in order to improve their accuracy. Tabular representation of the results are supplied in the appendix, variance figures are not supplied.

The remaining tests were carried out on a 33 MHz 80486 running System V 3.2. The drive was a Wren 5 with synchronous SCSI. The System V and B5FS file system tests were run together with the same hardware and the same kernel.

* Steven Bodnar is currently employed by Digital Ideas Pty Ltd.
† Stephen Prince is currently employed by Chancery Lane Computer Services Pty. Ltd.
‡ whilst this is not really a true indication of file system performance, it was deemed necessary in order to provide a consistent test environment.

Problems with comparing the performance of the System V file system with the Berkeley file system is that both are not available on most hardware. Tuning parameters cannot be easily mapped between System V and Berkeley operating systems. Other optimizations in the Berkeley operation system might colour the file system's throughput.

With the B5FS filesystem, the one System V operating system is used. Inode caching, symbolic links and large directory names are not supported, just the standard System V directory structure. Built into the B5FS file system was a method of switching on or off any of the optimizations using additional arguments to *tunefs*. This increased the accuracy of the comparisons.

The cache for System V was 400 one kilobyte buffers and the cache for the B5FS filesystem was 50 eight kilobyte buffers. The number of one, two and four kilobyte buffers was kept small enough to have no effect but large enough not to cause any performance problems. The one, two and four kilobyte buffers are not as used as often because they are copied directly to eight kilobyte buffers when they need to grow.

3. Cached File Performance

The initial release of B5FS gave good performance for large file sizes and a consistent bandwidth for a range of file sizes. Unfortunately for file sizes small enough to reside entirely in the buffer cache the bandwidth was still below what was experienced under an S5 file system, as measured on the hybrid system.

Closer examination of the S5 and B5FS implementations have revealed some further optimizations which have been applied to improve performance.

3.1. Fragment handling oprimization

Analysis found that when writing small files a fragment at a time the performance of B5FS file system is much less than that of the System V filesystem. With the Berkeley Fast File System being added to System V where the buffer size is 1K, this is the norm rather than the exeption. BSD$_g$ systems always try writing in an optimal blocksize rather than the fragment size.

The reason for the poor performance is as follows. To reduce fragmentation, incomplete blocks (fragments) will be allocated from already partially allocated blocks to save space. This could be a problem when any of the two partial blocks want to grow, but is solved by relocating the growing fragment to a full block. For performance not to suffer the implicite assumption is that the partial block is at the end of the file and is only grown after the file is reopened. This assumption does not hold when ill-behaved programs want to write less than a block size. Assuming the program writes in fragments there is constant fragment allocation and reallocation throughout the growing of the file. These ill-behaved programs are assumend to be rare on a Berkeley system but on a System V system it is the norm. For a discussion on the fragment allocation and reallocation policies refer to [McKusi84a]

The first attempt was to give every fragment a new block. Speed improved dramatically but total file system fragmentation was unacceptable. The solution was on the close of a file to perform the normal fragment relocation policy on the last block if it was a fragment. One extra benefit was that fragment allocation is ignored when the file requires indirect blocks to represent it (after about the first 96 kilobytes in size on an 8k/1k filesystem), so for writing large files a fragment at a time, no fragment reallocation would ever be necessary.

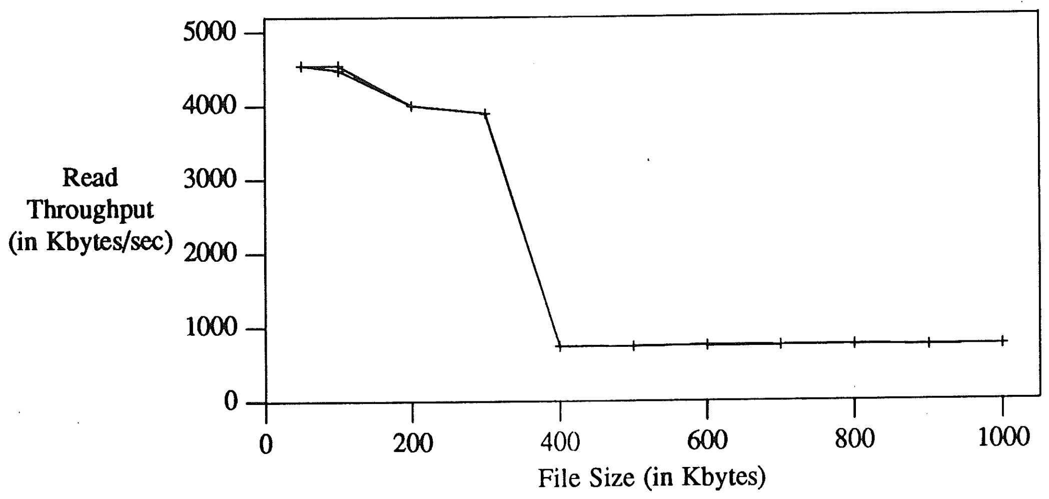
The results of adding the Fragmentation optimization can be seen in Figure 1 as the second lowest line, above the fully unoptimized B5FS line.

3.2. Buffer Reallocation Optimization

A change between 4.2 BSD‡ and 4.3 BSD releases was to reduce the amount of file system disk

† The BSD operating system comes from the Berkeley Standard Distribution of unix, distributed by the University of California at Berkeley.

‡ The 4.2 BSD and 4.3 BSD software are release 4.2 and 4.3 of the Berkeley Standard Distribution of unix.



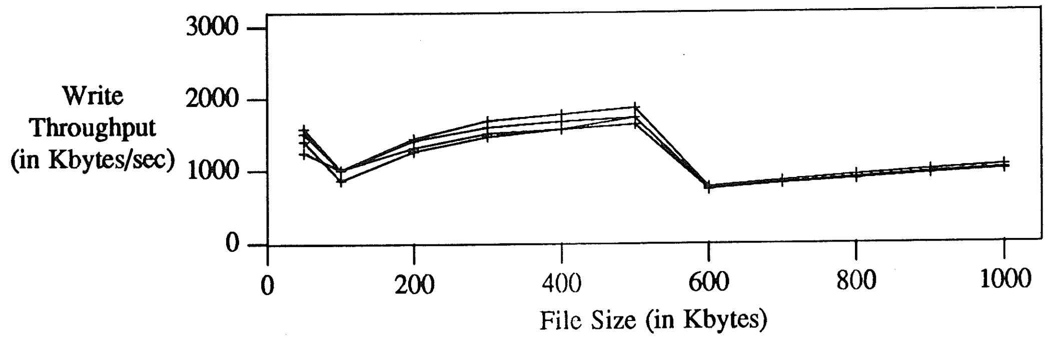


Figure 1: Incrementally adding optimizations. In all cases there is no degradation. Note that all data used a Write-Through policy. The lowest line, apart from the cross-over at the first point, is the totally unoptimized B5FS. The next line is with the Fragment optimization added, then the Buffer Reallocation optimization and finally the top line is with the Buffer Clearing optimization added. Refer to Table 1 in the Appendix for actual values.

block copying of data to successively larger and larger fragments until it finally grows to a full sized block. This is the result of a program slowly growing files using writes of 1024 bytes or less. The change was to recognize the first time that the file system is forced to copy a growing fragment and place it at the beginning of a full sized block. This optimization was applied to disk blocks but not to the buffer cache. The BSD system allocated system buffers that were only just large enough to hold the required portion of the block, and reallocate a buffer of this size as the block grows. This results† in a lot of block copying and hash queue searches every time the buffer is grown. An optimization to improve the write "time" efficiency, (as opposed to the existing "space" efficiency algorithm) is to recognize the first attempt to grow a buffer from the minimum class size to the next available size, and force it to allocate the new buffer as the maximum class size available.

The results of adding the Buffer Reallocation optimization can be seen in Figure 1 as the second highest line, just below the fully optimized B5FS line.

† according to the interpretation of results from profiler(1M) and bstat(1M). Bstat(1M) is a tool developed for collecting data related to buffer cache activity.

3.3. Buffer Clearing Optimization

A kernel profile of intense I/O on a B5FS file system, where all activity would remain in the buffer cache, we found that the most active procedure was iomove(), the copy from user space to kernel space. Second in line was bzero(). If you write 8k, an 8k buffer would be allocated, cleared, then overwritten with the 8k of user data. In this senario the limiting factor should only be memory to memory copy speeds. To further improve the cached performance we do not clear the buffer when reads and writes of valid data occur, only when reading from "a file-hole" or when dealing with directories and other important structures.

The results of finally adding the Buffer Clearing optimization can be seen in Figure 1 as the highest line.

3.4. Cache Write Policy

The purpose of buffer caching is to improve the average access time of disk blocks by keeping the most frequently used items in a small, fast, cache memory. The write policy determines when a modification is presented to secondary storage. Writes may always go directly to secondary storage (asynchronously) when using a *write-through* policy. Alternatively, the write may go to the cache to be written at some time later, usually when the buffer where the block is located is about to be replaced, using the *write-back* policy. A Write-back policy is motivated by the expectation that the block will be modified or accessed several times before it has to be written. Write-back, or *delayed-write* as it is referred to within the UNIX[†] kernel, is also desirable in file system caches because many files are temporary and may *never* have to be written [Thomps89a].

One of the major differences between a Berkeley file system and a System V file system, is that the buffer cache write policies are basically write-through and write-back respectively. In the Berkeley file system, if the write is a full block then it is immediately written asynchronously to disk, while only partial blocks are written with a delayed write. In the case of an System V file system the disk writes are via a delayed write.

3.4.1. Effects on Read Operations

When B5FS (as well as the Berkeley Fast File System) attempts to read a buffer that is currently being written to disk, it will block the reading process until it is available. Given a common senerio of one process writing a file and another reading the same file, a write-through strategy could result in the reader process waiting for the disk blocks to be asynchronously written. This can become the predominate bottleneck when the disk is too slow to service all the write requests by the time the reads come along. This would be seen as a read throughput similiar to that of reading directly from the disk. Although Figure 2 does not show this effect it has been seen on machines with slower disks.

A draw back with the write-back strategy is the cache can quickly fill with dirty buffers and then the system is forced to write them out syncronously. At this point a reader could also block waiting for the write to finish. This effect is shown in Figure 2 where the file starts to fill the cache and the read performance for write-back is below that of write-through, about a 300 kilobyte file in a 400 kilobyte cache. As this is a test that is repeated three times we can assume that the cache mainly consists of dirty blocks.

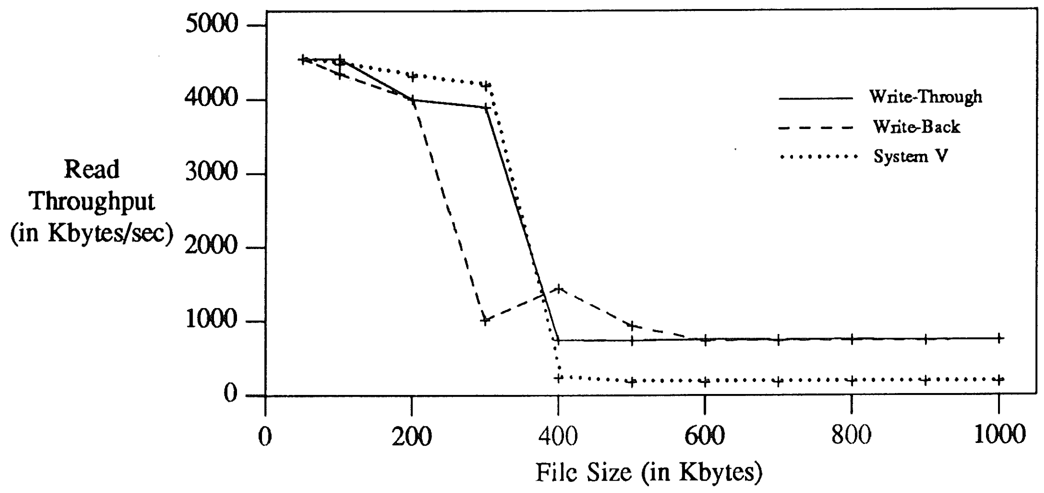
Finally the file can get so large that the reads never come across a block that is being written, as in the case of files much larger than the cache. Hence the read performance for both policies is equal, as shown in Figure 2.

3.4.2. Effects on Write Operations

The first difference we would expect to see is writes of files, that will fit in a cache, will be faster with write-back than with a write-through caching policy. The speed up is small but significant as the overhead of processing the start and finish of an asynchronous write is not negligible.[‡] There is a clear

[†] UNIX is a trademark of Bell Laboratories.

[‡] searching hash queues in the buffer pool, driver processing overhead.



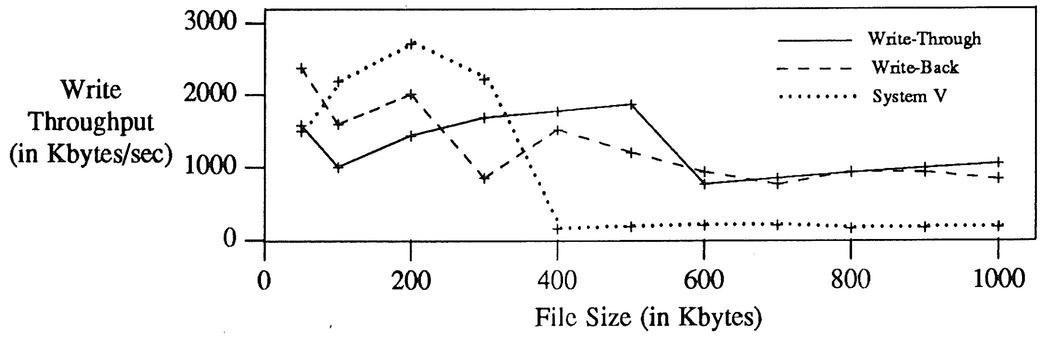


Figure 2: Comparison with System V and the fully optimized B5FS, with write-back and write-through cache policies. Refer to Table 2 in the Appendix for actual values.

example in Figure 2 with a 100 kilobyte file in a 400 kilobyte cache.

The drawback with the write-back strategy is when the file is much larger than the cache or the number of writes from other processes makes most of the buffers dirty. With the write-back case flushing dirty buffers to disk will not occur until the buffer is full, or the system decides to do a sync(2), then a large number of buffers will be suddenly queued. This is opposed to the write-through case where there is a more even distribution of writes to the disk. The overall effect is that write-back could dramatically increase the write times by first letting the disk be idle then swamping the disk with asyncronous writes. This results in a crossover between the write-back and write-through plots, as shown in Figure 2.

4. Conclusion

A test was performed first with no optimizations turned on in the B5 filesystem. Optimizations where then incrementally added one by one. First the Fragment optimization was enabled then the Buffer Reallocation optimization and finally the Buffer Clearing optimization. The results in Figure 1 show that there is an increase in performance for each addition of the optimizations. Although the optimizations yielded small but significant gains, they were more pronounced on much slower systems such as the National Semiconductor NS32332 series.

The choice of whether to have a write-through policy or a write-back policy is not clear and would greatly depend on the pattern of reads and writes. Two clear conditions emerge: a write-back cache is good if the traffic is mainly writes followed by reads on files small enough not to fill the cache (as would normally occur on /tmp) and a write-through policy is better if the traffic is mainly writes of large files in comparison to the cache with little or no probability of reading the file straight after, or even with just heavy multi-user use.

When B5FS is distributed all optimizations except Write-Back caching is enabled by default. Only in specific circumstances do we actually use the Write-Back policy of caching.

5. Further Research

Further work needs to be done on the negative sides of write-back and write-through strategies before one clearly out performs the other under all conditions. To improve the read after write problem the restriction to block on a write before reading the buffer could be lifted but work would have to be done on the consequences. To improve the swamping of the disk on writes with the write-back cache a more intelligent bdflush would need to be developed to take into account how many free buffers are left, possibly using high/low water marks on the number of buffers, disk bandwidth and even develope heuristics about the current process to determine when and how many buffers should be flushed to disk rather than when the buffer cache fills.

If you look at figure 1 or 2 the write-through strategy produces a dip in performance at about the 100 Kilobyte mark. It has been speculated that the busy-write buffers are the cause of this and could be due to a fault in our own buffer management implementation. More work is needed to explain and possibly correct this phenomenum.

6. Acknowledgements

We would like to thank Michael Podhorodecki and Russell McDonnel for allowing us to do this work and for supplying many ideas. Others who made helpful comments on the paper include Mark Tracey. We thank them all.

References

Prince88a.
Stephen Prince, "Porting the Berkeley Fast File System to System V," *AUUGN*, vol. 9, no. 6, AUUG Inc, December 1988.

McDone87a.
Ken J. McDonell, "Taking Performance Evaluation Out Of The "Stone" Age," *USENIX Summer Conference Proceedings*, Phoenix, Arizona, June 1987.

McKusi84a.
Marshall Kirk McKusick, William N. Joy, Samuel J. Leffer, and Robert S. Fabry, "A Fast File System for UNIX," in *Berkeley 4.3 BSD SMM*, p. 5, University of California, Berkeley, CA 94720, Febuary 18, 1984.

Thomps89a.
James G. Thompson and Alan Jay Smith, "Efficient (Stack) Algorithms for Analysis of Write-Back and Sector Memories," *TOCS*, vol. 7, no. 1, p. 78, ACM, February 1989.

APPENDIX

Write Throughput

File size (Kbytes)	None (Kbytes/sec)	Fragment (Kbytes/sec)	Fragment Buf Alloc (Kbytes/sec)	Fragment Buf Alloc Buf Clear (Kbytes/sec)
50	1402.1	1250.0	1515.2	1579.3
100	862.1	1010.1	1010.1	1010.1
200	1263.2	1314.0	1415.1	1445.8
300	1461.0	1509.9	1595.7	1688.6
400	1564.5	1568.6	1678.3	1775.2
500	1634.0	1730.1	1730.1	1863.4
600	735.3	745.1	750.0	770.9
700	810.2	821.0	821.0	851.6
800	870.6	887.0	887.3	924.9
900	936.5	942.8	953.4	990.1
1000	990.1	995.7	1006.0	1050.0

Read Throughput

File size (Kbytes)	None (Kbytes/sec)	Fragment (Kbytes/sec)	Fragment Buf Alloc (Kbytes/sec)	Fragment Buf Alloc Buf Clear (Kbytes/sec)
50	4545.4	4545.4	4545.4	4545.4
100	4479.6	4545.4	4545.4	4545.4
200	4000.0	4000.0	4000.0	4000.0
300	3896.1	3896.1	3896.1	3896.1
400	733.5	735.8	731.7	736.3
500	729.3	730.7	733.5	731.4
600	742.3	743.8	741.1	744.1
700	737.1	738.4	740.8	738.9
800	747.2	745.1	747.0	745.8
900	732.5	730.7	733.1	735.1
1000	738.7	740.0	742.0	741.3

Table 1: Incrementally adding optimizations.

Write Throughput

File size (Kbytes)	Write-through (Kbytes/sec)	Write-back (Kbytes/sec)	System V (Kbytes/sec)
50	1579.3	2380.9	1503.7
100	1010.1	1595.8	2190.0
200	1445.8	2010.6	2716.4
300	1688.6	855.2	2218.2
400	1775.2	1523.3	152.6
500	1863.4	1204.4	177.4
600	770.9	935.4	199.6
700	851.6	766.0	202.2
800	924.9	935.7	158.5
900	990.1	928.1	166.8
1000	1050.0	836.3	172.6

Read Throughput

File size (Kbytes)	Write-through (Kbytes/sec)	Write-back (Kbytes/sec)	System V (Kbytes/sec)
50	4545.4	4545.4	4545.4
100	4545.4	4347.8	4479.6
200	4000.0	4000.0	4317.0
300	3896.1	1011.2	4186.2
400	736.3	1441.2	233.4
500	731.4	930.1	171.0
600	744.1	725.8	174.0
700	738.9	724.7	171.4
800	745.8	733.3	174.7
900	735.1	732.1	175.9
1000	741.3	740.0	176.1

Table 2: Comparison with System V and the fully optimized B5FS, with write-back and write-through cache policies.

Open Buzzwords and NPAs

Greg Rose
Stephen Frede

Softway Pty Ltd

ABSTRACT

In recent times the computer industry has been moving towards increasingly "open" specifications. This has been evident in computer architectures, bus structures, and particularly operating systems and networks. The last bastion to fall appears to be that of trademarks, terminology and acronyms. This paper explores current efforts to bring these facets of the computer industry into the open, where a user friendly standards oriented approach can yield productivity gains in the standardisation of *de Jure* terminology rather than the current *de facto* usage.

1. Open Buzzwords

Regrettably, in the past, the computer industry has abounded with jargon terms, also known as buzzwords, which have generally been associated with particular hardware of software vendors. This practice must stop. Users are beginning to demand, in conjunction with Open Systems, the use of Open Buzzwords. That is, specific terminology for use in the computer industry which is of a non-proprietary nature. By far the most important enabler for this new technology is the NPA.

2. NPAs

An NPA is a vital part of the Opening of Buzzwords. An NPA is a Non-Proprietary Acronym, as opposed to a TLA[1] or Three Letter Acronym. TLAs appear to have been created by IBM[2] with such momentous successes as AMD[3] (Air Movement Device, more commonly known as a FAN, which is not an acronym).

TLAs have been such momentous buzzword enablers that other computer manufacturers have followed the *de facto* standard set thereby. Some examples which come to mind are CDC[4], ICL[5], CCI[6], and of course DEC[7].

Of course, there have been holdouts from this *de facto* standard. AT&T[8] not only added a fourth character, but it isn't even alphanumeric. DG[9] continued this divergence from their arch rivals by having a more ergonomic, shorter acronym; regrettably (for them) the market placehas moved in the opposite direction, towards acronyms like WYSIWYG (What You See Is What You Get), self-evidently much more user friendly. This is opposed to the application itself, which is more likely to be WYSINWYW (... Not What You Want).

1. TLA is perceived to be a trademark of International Business Machines
2. IBM is perceived to be a trademark of International Business Machines
3. AMD is perceived to be a trademark of International Business Machines
4. CDC is perceived to be a trademark of Control Data Corporation
5. ICL is perceived to be a trademark of Internation Computers Limited
6. CCI is perceived to be a trademark of Computer Consoles Incorporated
7. DEC is perceived to be a trademark of Digital Equipment Corporation
8. AT&T is perceived to be a trademark of American Telephone and Telegraph
9. DG is perceived to be a trademark of Data General

ETA[10] are (were) active by having a name which emulates a TLA without the drawback of needing an explanation for it.

In the early 1980's Zilog (not themselves an acronym) attempted to establish proprietary status for the SCA "Z", however courts ruled that degenerate acronyms are really just letters, and as such could not be trademarked. This was the first shot in the war for Buzzword Openness, establishing forever that there were buzzwords which could never be proprietary.

The fight against the *de facto* TLA standard has been entered by [11], but unfortunately they are not truly advocates of NPAs. A court ruling is expected soon in the suit brought by [12] alleging restraint of trade.

There is a schism between the two major factions attempting to establish *de facto* standards in the process of opening buzzwords. AI (Acronyms International) are in favour of pronouncable acronyms. Their name is, itself, a cry for assistance in this effort. Opposing them are the OSF (Overuse of Sibilants Forbidding pronunciation), who validly claim that of the 17576 possible TLAs, approximately 16473 are in use and only a small proportion of the others can be pronounced. Regrettably, there is internal dissent in the OSF regarding just how many are pronouncable, with the Australians asserting that all of them are[13], including OSF, while the European members believe that there are many more than 17576 in the first place.

The various Asian manufacturers, particularly the Japanese, seem to be remaining neutral in this conflict, since they are not worried about the possibility of running out of product descriptions.

10. ETA is perceived to be a trademark of Control Data Corporation
11. is perceived to be a trademark of Invisible Acronyms Backspace Backspace Corporation
12. is perceived to be a trademark of Nonproprietary Nonvisible Acronyms Backspace Backspace Backspace Limited
13. Aussies have been seen to pronounce things like WYSIAYG (What You See Is All You Get), and even command names like *fsck(1m)*.

Bulgeria In Turmoil

Greg Rose

Softway Pty Ltd

It took me some time to understand the reaction I was seeing in the face of the Bulgarian Computer Technologists when we discussed possible projects. Translated into words, it went "This sounds exciting; oh yes, I am allowed to be excited now..."

Bulgaria has been a controlled country for most of its history, and a burning hatred for the Turks, who held control for five centuries, is the most obvious result. The Russians liberated Bulgaria from the Turks and left it free for about 50 years, until first Nazi Germany asserted some control; and then communist rule took over. Strangely, the hatred for Communism is to some extent divorced from the Russians who brought it.

Despite the definite hate of Communism, the first free elections for 45 years returned the Communist Party with a slim majority, and both international observers and the Bulgarians agree that the election process was fair. However, the Bulgarians feel that the campaigning was definitely not, with the Communist Party exercising too much control over the media. There has been one unexpected disaster resulting from this election - Bulgaria, alone among the recently freed satellite states, is still "a communist country" and as such cannot receive most forms of foreign aid..

The Soviets are not really helping either. The week before I arrived the supply of crude oil was cut off with no reason given. In retrospect, it is obvious that the USSR wants to increase its export of oil in return for hard currency, rather than give it to Bulgaria for soft currency. This is interesting because Bulgaria used to import more than it needed, refine it, then export some of it for hard currency itself. This important source of trade cash has now dried up as the USSR will now export only as much as it thinks Bulgaria needs, and starting next year all trade between the old Communist countries will be conducted only in hard currencies.

[Note added during revision: I find the current middle east crisis, coming only three weeks after the Soviet Union plays games like this, to be highly suspicious - rising oil prices are likely to help the USSR significantly.]

Bulgaria had almost all of the Eastern Bloc computer market, for PC's, disk drives, IBM compatible mainframes and DEC compatible minis. About 15 years ago they succeeded in driving out of business the other East German and Russian manufacturers. This allowed them to begin working the monopoly the easy way, letting quality slide for example, and the political changes which are opening the purchasing options of the USSR are unmitigated disaster for the Bulgarian computer industry of today. Attempts to address these issues a few years ago were squashed as "unnecessary".

I was invited, through a contact in the Supercomputer Industry in the United States, to come to Bulgaria and talk to people about the growing importance of Unix, and to bring a personal view of the standards efforts and political machinations. The people there are so eager to read about changes in the Western computer industry that they believe everything they read, whether it is technical specifications, new benchmark figures, or downright marketing hype.

I arrived on Balkan Airways about three hours late, which is fairly normal for flights over Europe with its congested airways. Heathrow is not a fun place when two out of every three flights are delayed. The aircraft was a Tupelov TU-143, somewhat like a Boeing 727. In the seat pocket was a safety card for a TU-133 (DC9 like) - I think I was the only person who noticed, although all I saw were wrong.

I was met at the airport by Trayan Velitchkov, a Senior of Research at the Bulgarian Academy of Sciences. He started his car, a 13 year old Lada (Fiat 1500 clone) with 230,000km on it, and took me to a "hotel", which was really a private home with a couple of flats attached. The Lada had no windscreen wipers (I later found out that it is standard practice to remove the blades and lock them in the car because they are in short supply), and it wouldn't idle at all. Trayan was expert at revving the engine while braking with his left foot, or heel-and-toeing to keep it going.

This was also the only time I saw his car start itself. I became very proficient at pushing during the next 6 days. I arrived on Monday and on Saturday the Lada died.

Trayan and another researcher, Dimitri, were my hosts during the trip. The owner of the "hotel", Nadia, was my surrogate mother. Dimitri's car wouldn't start either, although I think this was partly because of the lack of use during the petrol shortage - he certainly knew nothing about clutch starting it, and I got all my experience driving Russian cars by doing this for him.

Sofia is Bulgaria's capital, and has less than 1 million of the 9 million people in the country. Other cities, which I didn't see, are Plovdiv, Varna (on the shore of the Black Sea) and Stara Zagora (Stara means old). Most of the living is high-rise, huge buildings with huge advertisements on the featureless side walls.

Most buildings I saw were more or less in a state of disrepair, although some were very elegant stone and spacious architecture. Except for agriculture, no effort seemed to go into gardening. I didn't see any mown grass anywhere on the trip.

Just about everyone I met, almost all technical people, programmers or hardware designers, asked about how to migrate to Australia, or just about anywhere. The Turkish occupation and communist rule have left them with a very defeatist attitude. To be fair, it is hard to see how Bulgaria will get out of the mess it is in, but most people don't want to try.

The defeatist attitude is manifested, technologically, as a lack of confidence to do design. The Bulgarians have very successfully cloned the IBM 370 series, complete with channels, comms devices, and disk drives, as well as the DEC VAX 11/750 (although faster), and PC's. They also have Floating Point Systems compatible array processors at 18Mflops each, and are designing a workstation (for the Eastern Block) marrying a 386 PC and an array processor. While the implementations are good, fast and innovative, the design was all done elsewhere, and the Bulgarians are reluctant to do anything from scratch.

In software, too, they are used to doing patches and maintenance, but not large projects or innovative software design. The academies seem to support the old notion that software is just an add-on, to make the hardware sit up.

Currently, the only experience any of them have with Unix has been to get it running on the 370 compatible machines, and then move that to the VAX. They started with a version of Amdahl's UTS, which they freely admit was stolen. Just how it was stolen, nobody knows, as "procurement" was one of the questions they didn't ask. It will be hard for them to use this, though, as Bulgaria is now a signatory to the Berne Convention for intellectual property. The hard currency for buying a Unix licence from AT&T is almost certainly not available in the near future, so they are looking hard at the Free Software Foundation and Berkeley's new release.

USENIX Association News For AUUG Members

Donnalyn Frey
donnalyn@frey.com
Frey Communications

Donnalyn is the USENIX Association Press Liaison. She provides members of the press, USENIX Association members, and AUUG members with information on the activities of the USENIX Association.

1. The 1990 Summer USENIX Association Conference

Dennis Ritchie, of AT&T Bell Laboratories and co-author of the UNIX operating system, presented the keynote address at the USENIX Association 1990 Summer Technical Conference and Exhibition on June 11 - 15 at the Anaheim Marriott Hotel and Convention Center in Anaheim, California. He reflected on "What Happens When Your Kid Turns 21?", reported here by Marc Donner.

OPEC vs. the Medellin Cartel

Dennis Ritchie, looking distinguished in a rarely worn suit and tie, delivered the USENIX Conference and Exhibition keynote address entitled "What Happens When Your Kid Turns 21?" on Wednesday, June 13. This was Dr. Ritchie's first conference address in over three years. *[Has it really been that long since he spoke at AUUG'89? -Ed.]* Departing from conventional expository style, Dr. Ritchie's keynote was in the form of a telephone interview.

Ritchie began by acknowledging the influence of Multics, a project on which both he and Ken Thompson had participated in the 1960s. Features of UNIX inspired by Multics include the tree-structured file system and the concept of the shell as a separate program. He mentioned that he is a bystander in UNIX and C today.

Ritchie answered questions on the current state of UNIX and C, particularly recent standardization efforts. He noted that the ANSI C standard seems to be a good standard, having clarified and updated many items without breaking them. In addition, he observed that while the core of the POSIX effort seemed sound, he was uncomfortable with the lack of coherent network facilities in the proposed standard and he found Real-Time Extensions work to be troublesome. In general, he mentioned he was ambivalent about standards as they seem to create a tension between standardization and innovation. The real problem, he noted, is when is the time right to standardize?

Ritchie briefly commented on the contest between the Open Software Foundation and UNIX International, observing that the squabbling was harmful but, given the development history of UNIX, inevitable. The future movie could be called "OPEC vs. the Medellin Cartel" or perhaps "Ninja Turtles vs. the Fantastic Four". He mentioned that when he told Ken Thompson (on sabbatical in Australia) about the OSF, Ken noted dryly, "Imagine, IBM and DEC in the same room, and we did it." Ritchie also commented that the OSF could be criticized for not producing less than might be expected, but agreed that the bygone days of easily available source code were definitely better.

The imaginary interviewer on the telephone asked Ritchie his opinion of C++. Ritchie declined to comment, citing an "agreement with Bjarne [Stroustroup] that I don't give lectures on C++ and he doesn't talk about old C."

The final part of the keynote "interview" was more philosophical in tone. Asked about his greatest satisfaction from the UNIX work, Ritchie cited the influence it has had in creating new companies and new directions for old companies. The advent of inexpensive high-performance microprocessors was brought about, in his view, by the wide availability of portable software to run on them, eliminating the need for an expensive software development effort.

Asked about things he would do differently, Ritchie responded that Ken [Thompson] always said he would spell 'creat' with an 'e.' "More seriously," he said, "there are lots of small sins." He mentioned the experiment in declaration syntax embodied in C had probably not succeeded. He complained that many implementations of UNIX had become bloated, observing that the 10th Edition, the latest version at Bell Laboratories, required only 140K of text space. Overall, Ritchie said, "I have very few qualms about UNIX as a whole; it has turned out to be very adaptable."

When asked to discuss his regrets, Ritchie noted that the opportunity to create a viable alternative to X was lost when the Blit work was not aggressively pursued. He also regretted that the UNIX work had not satisfied AT&T's need for a system to manage very large software development, like the 5ESS system. He mentioned that either the small-is-beautiful development model of simple, elegant powerful tools isn't appropriate to large system development, or the message didn't get across.

Asked for a message to attendees, Ritchie observed that UNIX, as a child that has come of age, was independent and that all he could do is let it run its own life and wish

This was also the only time I saw his car start itself. I became very proficient at pushing during the next 6 days. I arrived on Monday and on Saturday the Lada died.

Trayan and another researcher, Dimitri, were my hosts during the trip. The owner of the "hotel", Nadia, was my surrogate mother. Dimitri's car wouldn't start either, although I think this was partly because of the lack of use during the petrol shortage - he certainly knew nothing about clutch starting it, and I got all my experience driving Russian cars by doing this for him.

Sofia is Bulgaria's capital, and has less than 1 million of the 9 million people in the country. Other cities, which I didn't see, are Plovdiv, Varna (on the shore of the Black Sea) and Stara Zagora (Stara means old). Most of the living is high-rise, huge buildings with huge advertisements on the featureless side walls.

Most buildings I saw were more or less in a state of disrepair, although some were very elegant stone and spacious architecture. Except for agriculture, no effort seemed to go into gardening. I didn't see any mown grass anywhere on the trip.

Just about everyone I met, almost all technical people, programmers or hardware designers, asked about how to migrate to Australia, or just about anywhere. The Turkish occupation and communist rule have left them with a very defeatist attitude. To be fair, it is hard to see how Bulgaria will get out of the mess it is in, but most people don't want to try.

The defeatist attitude is manifested, technologically, as a lack of confidence to do design. The Bulgarians have very successfully cloned the IBM 370 series, complete with channels, comms devices, and disk drives, as well as the DEC VAX 11/750 (although faster), and PC's. They also have Floating Point Systems compatible array processors at 18Mflops each, and are designing a workstation (for the Eastern Block) marrying a 386 PC and an array processor. While the implementations are good, fast and innovative, the design was all done elsewhere, and the Bulgarians are reluctant to do anything from scratch.

In software, too, they are used to doing patches and maintenance, but not large projects or innovative software design. The academies seem to support the old notion that software is just an add-on, to make the hardware sit up.

Currently, the only experience any of them have with Unix has been to get it running on the 370 compatible machines, and then move that to the VAX. They started with a version of Amdahl's UTS, which they freely admit was stolen. Just how it was stolen, nobody knows, as "procurement" was one of the questions they didn't ask. It will be hard for them to use this, though, as Bulgaria is now a signatory to the Berne Convention for intellectual property. The hard currency for buying a Unix licence from AT&T is almost certainly not available in the near future, so they are looking hard at the Free Software Foundation and Berkeley's new release.

USENIX Association News For AUUG Members

Donnalyn Frey
donnalyn@frey.com
Frey Communications

Donnalyn is the USENIX Association Press Liaison. She provides members of the press, USENIX Association members, and AUUG members with information on the activities of the USENIX Association.

1. The 1990 Summer USENIX Association Conference

Dennis Ritchie, of AT&T Bell Laboratories and co-author of the UNIX operating system, presented the keynote address at the USENIX Association 1990 Summer Technical Conference and Exhibition on June 11 - 15 at the Anaheim Marriott Hotel and Convention Center in Anaheim, California. He reflected on "What Happens When Your Kid Turns 21?", reported here by Marc Donner.

OPEC vs. the Medellin Cartel

Dennis Ritchie, looking distinguished in a rarely worn suit and tie, delivered the USENIX Conference and Exhibition keynote address entitled "What Happens When Your Kid Turns 21?" on Wednesday, June 13. This was Dr. Ritchie's first conference address in over three years. *[Has it really been that long since he spoke at AUUG'89? -Ed.]* Departing from conventional expository style, Dr. Ritchie's keynote was in the form of a telephone interview.

Ritchie began by acknowledging the influence of Multics, a project on which both he and Ken Thompson had participated in the 1960s. Features of UNIX inspired by Multics include the tree-structured file system and the concept of the shell as a separate program. He mentioned that he is a bystander in UNIX and C today.

Ritchie answered questions on the current state of UNIX and C, particularly recent standardization efforts. He noted that the ANSI C standard seems to be a good standard, having clarified and updated many items without breaking them. In addition, he observed that while the core of the POSIX effort seemed sound, he was uncomfortable with the lack of coherent network facilities in the proposed standard and he found Real-Time Extensions work to be troublesome. In general, he mentioned he was ambivalent about standards as they seem to create a tension between standardization and innovation. The real problem, he noted, is when is the time right to standardize?

Ritchie briefly commented on the contest between the Open Software Foundation and UNIX International, observing that the squabbling was harmful but, given the development history of UNIX, inevitable. The future movie could be called "OPEC vs. the Medellin Cartel" or perhaps "Ninja Turtles vs. the Fantastic Four". He mentioned that when he told Ken Thompson (on sabbatical in Australia) about the OSF, Ken noted dryly, "Imagine, IBM and DEC in the same room, and we did it." Ritchie also commented that the OSF could be criticized for not producing less than might be expected, but agreed that the bygone days of easily available source code were definitely better.

The imaginary interviewer on the telephone asked Ritchie his opinion of C++. Ritchie declined to comment, citing an "agreement with Bjarne [Stroustroup] that I don't give lectures on C++ and he doesn't talk about old C."

The final part of the keynote "interview" was more philosophical in tone. Asked about his greatest satisfaction from the UNIX work, Ritchie cited the influence it has had in creating new companies and new directions for old companies. The advent of inexpensive high-performance microprocessors was brought about, in his view, by the wide availability of portable software to run on them, eliminating the need for an expensive software development effort.

Asked about things he would do differently, Ritchie responded that Ken [Thompson] always said he would spell 'creat' with an 'e.' "More seriously," he said, "there are lots of small sins." He mentioned the experiment in declaration syntax embodied in C had probably not succeeded. He complained that many implementations of UNIX had become bloated, observing that the 10th Edition, the latest version at Bell Laboratories, required only 140K of text space. Overall, Ritchie said, "I have very few qualms about UNIX as a whole; it has turned out to be very adaptable."

When asked to discuss his regrets, Ritchie noted that the opportunity to create a viable alternative to X was lost when the Blit work was not aggressively pursued. He also regretted that the UNIX work had not satisfied AT&T's need for a system to manage very large software development, like the 5ESS system. He mentioned that either the small-is-beautiful development model of simple, elegant powerful tools isn't appropriate to large system development, or the message didn't get across.

Asked for a message to attendees, Ritchie observed that UNIX, as a child that has come of age, was independent and that all he could do is let it run its own life and wish it well.

With that message, the interviewer asked Ritchie what he had been doing lately? After mentioning having written an ANSI preprocessor for C, Ritchie proceeded to show a videotape of an infamous practical joke that Ritchie, Rob Pike, and magicians Penn and Teller played on their boss, Nobel Prize winner Arno Penzias.

1.1 Technical Exhibition

The Technical Exhibition included over 63 hardware and software companies displaying their latest technical innovations to a high focused end user community. Some of the participating exhibitors included IBM, Data General, AT&T, Intergraph, Sequent, Hewlett-Packard/Apollo, Digital Equipment Corp., Sequoia, Amdahl, Sun Microsystems, UUNET Communications, UNIX International, Open Software Foundation, NeXT, and HCR Corp.

The Association again sponsored an Ethernet network, which allowed exhibitors to display the networking capabilities of their products. Sun Microsystems and MIPS also used the Association's exhibition FDDI network.

1.2 Concurrent Sessions

A second track of the conference sessions featured informal talks on computing subjects. The most popular session was on computer generated music. Peter Langston of Bell Communications Research and Mike Hawley of MIT Media Lab showed the audience how computers make music, and how are they being used in music production, arrangements, and composition. The concurrent sessions also included Andrew Hume repeating his popular talks on regular expressions and make; Craig Hunt, of the National Institute of Standards and Technology discussing TCP/IP system administration; and Rob Kolstad of Sun Microsystems moderating a system administration problem solving panel. Neil Groundwater discussed XXXXX *[sic.]*

1.3 The Terminal Room and FaceSaver at the Conference

The USENIX Association hosted a Terminal Room with modems for a dialout connection and a T-1 connection to CERFnet and the Internet. Conference attendees could log onto their home or work systems to read their mail and contact other UNIX users directly from the conference.
The FaceSaver faces from the Anaheim conference will again go to the UUNET FaceServer.

2. 1991 Winter Conference in Dallas, Texas

The 1991 USENIX Association Winter conference is in Dallas, Texas on January 21 - 25, 1991. The Summer 1990 conference had a theme which was retrospective in nature. For this conference we once again look to the future. The theme of the 1991 Winter conference is "What's next: by the year 2010, evolution or revolution? Unix derivative or Something Else?"

Topics of papers at the conference may include:

Operating systems of the future:
- Distributed Systems
- Real-time Systems
- Object Oriented Systems
- Fault Tolerant Systems
- Multiprocessor and Multicomputer Systems
- Workstation Systems
- Systems for Novel Architectures

Communications and Networking:
- Protocols
- Performance
- Administration
- Security

Applications:
- Databases
- Transaction Processing
- Arts and Social Applications
- Novel Application Areas

User Interfaces:
- Human Factors
- Graphics and Window Systems
- Graphical User Interfaces

Programming Environments and Languages

Testing and Debugging

To request additional technical information, please contact:

Lori S. Grob
Dallas USENIX Technical Program
Chorus systemes
6, avenue Gustave Eiffel
F78182 Saint-Quentin-en-Yvelines CEDEX France

Internet:	dallas-conf@usenix.org
UUCP:	uunet!usenix!dallas-conf
Telephone:	+33 (1) 30 57 00 22
FAX:	+33 (1) 30 57 00 66

Please include your physical and electronic mail address in all correspondence.

For information on attending the conference, please contact the USENIX conference office.

3. Monograph Series

Marc Donner, of the IBM Thomas J. Watson Research Center, has been selected as the editor of the USENIX Association's Monograph Series on Advanced Computing Systems, published by the USENIX Association.

The USENIX Association intends to publish books and monographson the general topic of computing systems. The intended audience for these books is the community of system designers, builders, users, and scholars. The intent is to publish material of lasting interest and importance, with an emphasis on actual systems. Subjects may include design, implementation, history, and analysis of real systems. While Marc is interested in UNIX and UNIX-inspired systems, he does not expect to limit his attention to such systems.

Marc would like to solicit manuscripts in two specific areas - books in traditional styles and formats about topics important to the systems community as well as things new or unusual.

Among things new or unusual, Marc is interested in exploring significant systems, code, and important technical reports.

Significant systems - many significant systems are documented, if at all, only in reference manuals or user guides. Journal publications often concentrate on narrow specific details, as is appropriate for focussed technical audiences. What is lost is the broad description of the design and its evolution, with consideration of the success and failure of specific features and lessons learned.

Code - The editor is interested in exploring the possibilities of publishing code to read. A truism among the programming community is that one learns to write good programs by reading good and bad programs. Sadly, there is little code available to read. The recent interest in public-domain code and open systems has increased the quantity of high-quality source code available. Many open questions in the publication of code remain to be explored. The conventional codex form, long accepted as appropriate for literary works and texts, may not be the right one for programs. Very few experiments have been made with this form, something that I hope to encourage. The audience for published code includes serious students of systems, including both the undergraduate and advanced levels, and practitioners involved with development, modification, and analysis of actual systems.

Important technical reports - many important technical reports, issued in small numbers by industrial organizations, research labs, or university departments, are not disseminated as widely as they merit. This is often because the originating organization doesn't have the resources or the will to publish it more widely and because the material is deemed inappropriate by commercial publishers because of its narrow scope or limited size. Many technical reports are too large for journal publication and too small for conventional book publication. Marc hopes to provide a means of publication and distribution of the best of these.

To submit a manuscript or proposal for consideration for the MonographSeries, send a copy to

Monograph Editor
USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

or send electronic mail to

monographs@usenix.org

4. Computing Systems Music CD

The USENIX Association is pleased to present the latest issue of *Computing Systems*, "A Musical Offering," complete with a compact disk containing all the musical illustrations associated with the papers in this issue. Original computer music will be presented by Peter Langston and Mike Hawley, with accompanying papers to discuss the music. Tim Thompson includes a paper on Keynote, a language and extensible graphic editor for music. Lastly, the issue includes a controversy discussion on portability by Stuart Feldman and W. Morven Gentleman. The authors, Peter Langston, Mike O'Dell, editor, and Peter Salus, managing editor of *Computing Sytems*, have worked for over a year on this issue. It should be one of the most interesting issues of *Computing Systems* to date.

5. 1990 USENIX Workshops

Upcoming workshops include:

Mach on October 4 - 5 at the Radisson Hotel in Burlington, Vermont.

Large Installation Systems Administration IV on October 17 - 19 in Colorado Springs, Colorado.

Software Development Environments in UNIX on January 16 - 18, 1991 at Grand Kempinski Hotel in Dallas, Texas cosponsored with the SIGMA Project of Japan.

Distributed / Multiprocessor Systems Symposium, co-sponsored with SERC of Purdue University on March 21 - 22, 1991 in Atlanta, Georgia.

Contact the USENIX conference office for information on these workshops.

6. Speakers Bureau

A Speakers Bureau was recently begun to provide a forum for people with expertise in various areas of UNIX and advanced computing to share their knowledge with educational groups, including high schools, colleges, universities, and local user groups. Potential speakers have been encouraged to contact the USENIX office for more information.

7. Further Information on Conferences and Workshops

If you need further information regarding USENIX conferences or workshops, contact the USENIX Conference Office at:

22672 Lambert Street
Suite 613
El Toro, CA 92630
USA

Email: judy@usenix.org or {uunet,ucbvax}!usenix!judy
Tel: +1 714 588 8649
FAX: +1 714 588 9706

8. Further Information about the USENIX Association

If you would like information on membership, or would like information on ordering USENIX publications (proceedings,manuals, the technical journal, Computing Systems, the Monograph Series, or the Association's newsletter, ;login:, please contact the USENIX Association Executive Office at:

2560 Ninth Street
Suite 215
Berkeley, CA 94710
USA

Email: office@usenix.org
Tel: +1 415 528 8649
FAX: +1 415 548 5738.

AUUGN Back Issues

Here are the details of back issues of which we still hold copies. All prices are in Australian dollars and include surface mail within Australia. For overseas surface mail add $2 per copy and for overseas airmail add $10 per copy.

pre 1984	Vol 1-4	various	$10 per copy
1984	Vol 5	Nos. 2,3,5,6	$10 per copy
		Nos. 1,4	unavailable
1985	Vol 6	Nos. 2,3,4,6	$10 per copy
		No. 1	unavailable
1986	Vol 7	Nos. 1,4-5,6	$10 per copy
		Nos. 2-3	unavailable
			(Note 2-3 and 4-5 are combined issues)
1987	Vol 8	Nos. 1-2,3-4	unavailable
		Nos. 5,6	$10 per copy
1988	Vol 9	Nos. 1,2,3	$10 per copy
		Nos. 4,5,6	$15 per copy
1989	Vol 10	Nos. 1-6	$15 per copy
1990	Vol 11	No. 1,2	$15 per copy

Please note that we do not accept purchase orders for back issues except from Institutional members. Orders enclosing payment in Australian dollars should be sent to:

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

WAUG
Western Australian UNIX systems Group

PO Box 877, WEST PERTH 6005

Western Australian Unix systems Group

The Western Australian UNIX systems Group (**WAUG**) was formed to bring together people with a common interest in UNIX systems. It provides a forum for exchange of ideas and experience among its members, catering for all levels of experience, from novice to guru, and the many UNIX and UNIX-like systems and supporting hardwares.

A major activity of the group is monthly meetings. Invited speakers address the group on topics including new hardware, software packages and technical dissertations. After the meeting, we gather for refreshments, and an opportunity to informally discuss any points of interest. Formal business is kept to a minimum.

The group also produces a periodic Newsletter, **YAUN** (Yet Another UNIX Newsletter), containing members contributions and extracts from various UNIX Newsletters and extensive network news services. **YAUN** provides members with some of the latest news and information available.

Meetings are held on the third Wednesday of each month at 6pm. For further information about membership or meeting venues, please contact:

the membership secretary, Major (major@pyrmania.oz) on (09) 474 2600, or
the chariman, Glenn Huxtable (glenn@wacsvax.uwa.oz) on (09) 380 2878.

WAUG is a local chapter of the AUUG and welcomes AUUG members and visitors to our meetings.

Glenn Huxtable,
Chairman, WAUG

AUUG

Membership Categories

Once again a reminder for all ''members'' of AUUG to check that you are, in fact, a member, and that you still will be for the next two months.

There are 4 membership types, plus a newsletter subscription, any of which might be just right for you.

The membership categories are:

Institutional Member
Ordinary Member
Student Member
Honorary Life Member

Institutional memberships are primarily intended for university departments, companies, etc. This is a voting membership (one vote), which receives two copies of the newsletter. Institutional members can also delegate 2 representatives to attend AUUG meetings at members rates. AUUG is also keeping track of the licence status of institutional members. If, at some future date, we are able to offer a software tape distribution service, this would be available only to institutional members, whose relevant licences can be verified.

If your institution is not an institutional member, isn't it about time it became one?

Ordinary memberships are for individuals. This is also a voting membership (one vote), which receives a single copy of the newsletter. A primary difference from Institutional Membership is that the benefits of Ordinary Membership apply to the named member only. That is, only the member can obtain discounts an attendance at AUUG meetings, etc. Sending a representative isn't permitted.

Are **you** an AUUG member?

Student Memberships are for full time students at recognised academic institutions. This is a non voting membership which receives a single copy of the newsletter. Otherwise the benefits are as for Ordinary Members.

Honorary Life Membership is not a membership you can apply for, you must be elected to it. What's more, you must have been a member for at least 5 years before being elected.

It's also possible to subscribe to the newsletter without being an AUUG member. This saves you nothing financially, that is, the subscription price is greater than the membership dues. However, it might be appropriate for libraries, etc, which simply want copies of AUUGN to help fill their shelves, and have no actual interest in the contents, or the association.

Subscriptions are also available to members who have a need for more copies of AUUGN than their membership provides.

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

Check that your membership isn't about to expire (or worse, hasn't expired already). Ask your colleagues if they received this issue of AUUGN, tell them that if not, it probably means that their membership has lapsed, or perhaps, they were never a member at all! Feel free to copy the membership forms, give one to everyone that you know.

If you want to join AUUG, or renew your membership, you will find forms in this issue of AUUGN. Send the appropriate form (with remittance) to the address indicated on it, and your membership will (re-)commence.

As a service to members, AUUG has arranged to accept payments via credit card. You can use your Bankcard (within Australia only), or your Visa or Mastercard by simply completing the authorisation on the application form.

AUUG Incorporated
Application for Institutional Membership
Australian UNIX* systems Users' Group.

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

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, 1991

.... 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.3 binary
☐ System V.2 source	☐ System V.2 binary
☐ System V source	☐ System V binary
☐ System III source	☐ System III binary
☐ 4.2 or 4.3 BSD source	
☐ 4.1 BSD source	
☐ V7 source	

☐ Other *(Indicate which)* ....

AUUG Incorporated
Application for Ordinary, or Student, Membership
Australian UNIX* systems Users' Group.

*UNIX is a registered trademark of AT&T in the USA and other countries

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, 1991

I, ... do hereby apply for

☐ Renewal/New* Membership of the AUUG	$78.00	
☐ Renewal/New* Student Membership	$45.00	(note certification on other side)
☐ International Surface Mail	$20.00	
☐ International Air Mail	$60.00	(note local zone rate available)
Total remitted	AUD$______ (cheque, money order, credit card)	

* Delete one.

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

Date: __/__/__ Signed: ____________________

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

For our mailing database - please type or print clearly:

Name: .... Phone: .... (bh)

Address: .... (ah)

....

.... Net Address: ....

....

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

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

Account number: ______ ______ ______ ______. Expiry date: __/__.

Name on card: ______________________________ Signed: ____________________

Office use only:

Chq: bank ________ *bsb* ___-___ *a/c* ______________ # ______________

Date: __/__/__ *$* *CC type* ___ *V#* ___________

Who: ____________ *Member#* _________

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

I, ... certify that

.... *(name)*

is a full time student at .... *(institution)*

and is expected to graduate approximately ___/___/___ .

Title: ________________________________ Signature: ____________________

AUUG Incorporated
Application for Newsletter Subscription
Australian UNIX* systems Users' Group.

*UNIX is a registered trademark of AT&T in the USA and other countries

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, 1991

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 AT&T in the USA and other countries.

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: ....
....	
....	
....	

Office use only:

Date: ___/___/___

Who: _____________ *Memb#* _________